PROYECTO DE GRADO

SISTEMA DE ANALISIS TECNICO PARA ACTIVOS DE RENTA VARIABLE

POR

DANIEL LOAIZA PALACIO
COD 200020030010

ASESOR

HERNÁN DARÍO TORO

UNIVERSIDAD EAFIT

MEDELLIN

2007

TABLA DE CONTENIDO

CAPITULO 1: DESCRIPCION DE LA METODOLOGIA DEL PROYECTO DE GRADO

I. INFORMACION GENERAL DEL PROYECTO

- **Autor:**

Daniel Loaiza Palacio COD 200020030010

- **Titulo:**

Sistema de Análisis Técnico para Mercados de Renta Variable (SATMRV).

- **Línea de investigación:**

Mercado colombiano de renta variable con el apoyo de herramientas informáticas desarrolladas en el marco del mismo proyecto.

- **Tipo de proyecto:**

Desarrollo de aplicativo software.

II. RESUMEN DEL PROYECTO

Dentro de los mercados financieros se encuentra el mercado de títulos de renta variable, que es aquel en donde la rentabilidad de la inversión, está ligada a las utilidades obtenidas por la empresa en la cual se invirtió y por las ganancias de capital obtenidas por la diferencia entre el precio de compra y venta. Este mercado esta compuesto por acciones, derechos de suscripción y títulos provenientes de procesos de titularización donde la rentabilidad no esta asociada a una tasa de interés específica. En este trabajo se abordara el análisis del comportamiento de este mercado desde el punto de vista técnico es decir, el estudio de los movimientos de mercado como son el precio, el volumen y el interés abierto; principalmente mediante el uso de gráficos, con el propósito de pronosticar las futuras tendencias de los precios, basándose en la premisa de que cualquier factor que posiblemente pueda afectar el precio, ya sea razones fundamentales, políticas, psicológicas u otras, se refleja realmente en el precio de ese mercado.

El propósito de este trabajo consistirá en el desarrollo de una herramienta informática para el análisis técnico de los mercados de renta variable en el mercado de valores colombiano, para aprovechar el gran potencial que nos brindan los computadores a la hora de analizar grandes cantidades de información.

III. DESCRIPCIÓN DEL PROYECTO

- **Justificación**

Debido al precio cambiante de los activos de renta variable, y la gran cantidad de información a analizar para identificar tendencias probables en el mercado basado en indicadores de análisis técnico. Se hace necesario el desarrollo de una herramienta de software para el cálculo y graficación de dichos indicadores en el mercado accionario colombiano. Dentro de estos índices cabe destacar las velas japonesas, indicadores de tendencias y osciladores.

- **Marco teórico:**

Bolsa de valores de Colombia (BVC)

“En los últimos años la bolsa de valores de Colombia ha adquirido protagonismo Tras el *boom* sin precedentes del mercado de valores de los últimos dos años, la bolsa ha pasado de ser una entidad fragmentada y conocida sólo por una pequeña élite especializada, a ser un centro nacional de operaciones financieras que rara vez deja los titulares de los diarios.

Los orígenes de la bolsa se remontan a los años veinte, cuando la industrialización incipiente iniciada a finales del siglo XIX había creado el capital excedente y la necesidad práctica de un sistema financiero moderno.

Fue entonces cuando se establecieron en el país sucursales de la banca extranjera, se constituyeron varios bancos comerciales, el banco central (Banco de la República) y la Superintendencia Bancaria y se promulgó la legislación bancaria que habría de durar hasta 1990" (Molina 2002).

"La gestión de Augusto Acosta como presidente de la bolsa de valores entre 1997 y 2005 fue un período en el que la bolsa atravesó un proceso acelerado de modernización y racionalización burocrática. Cuatro procesos modernizadores explican el profundo cambio institucional de la bolsa en esta época:

1. Unificación nacional: La existencia de tres bolsas regionales era un obstáculo poderoso para la formación transparente y homogénea de los precios de los instrumentos financieros en el país, pues una misma acción se transaba a distintos precios en los tres nodos regionales, además había ineficiencias y no eran instituciones sólidas.

2. Conversión de un sistema personalista de transacciones a un sistema impersonal de negociación pública**:** La institución de la bolsa en Colombia fue concebida durante todo el siglo XX bajo el modelo de un espacio cerrado de negociación entre miembros de una élite altamente cohesionada, en las que las relaciones y contactos personales constituían el principal vehículo de negociación. La bolsa se convirtió, tras la consolidación de 2001, en el foro de negociación principal de instrumentos financieros del país. Por tanto, más allá de su misión de facilitar la financiación de empresas mediante acciones, la BVC presta un servicio más amplio de plataforma de negociación de múltiples instrumentos financieros, desde bonos hasta acciones y divisas. Como lo sostiene el director de operaciones de una de las firmas comisionistas más grandes del país, "el mercado ahora es ciego" en la medida en que compradores y vendedores interactúan de manera anónima a través de pantallas de computador donde se publica la información de los precios de compra y venta de los instrumentos financieros. Utilizando esta

información, los comisionistas y demás actores del mercado "pujan" y compiten en tiempo real por los instrumentos en negociación.

3. Actualización tecnológica: La "desmutualización" de la bolsa no habría sido posible sin la adopción de tecnologías que le permitieran funcionar, en tiempo real, como plataforma pública en la que múltiples actores transan de forma anónima instrumentos financieros.

4. Internacionalización: La liberalización financiera de los noventa implicó la apertura del mercado bursátil colombiano a los flujos financieros internacionales y a la repatriación de capitales nacionales que, violando la antigua legislación de control de cambios, estaban invertidos o depositados en el exterior. Como lo afirmó el Presidente de la BVC, "el mercado de capitales está globalizado" en el país, así lo muestran las cifras sobre la inversión extranjera en el mercado de valores colombiano, que pasó de niveles insignificantes en los noventa a 2.525 millones de dólares en 2001, 2.115 millones en 2002 y 1.801 en 2003, para luego ascender a niveles sin precedentes en los últimos años –coincidiendo con el nuevo *boom* de los llamados mercados emergentes—, hasta llegar a 3.005 millones en 2004 y 5.890 millones de dólares en 2005 (BVC 2006)."[1]

En la bolsa de valores de Colombia no es posible hacer ventas en corto. Esto es, que las firmas comisionistas que poseen acciones de muchas personas y compañías no pueden entregar en préstamo dichas acciones para que otros realicen operaciones con ellas buscando un beneficio económico. Es decir, las acciones no se pueden tomar prestadas y venderlas al precio que se encuentren en ese momento, con el objetivo de comprarlas a un precio más bajo nuevamente y así devolvérsela a la firma prestataria, y de este modo obtener una ganancia por la diferencia de dinero entre la venta y la compra. (Hernández 2005).

[1] Tomado de http://cmd.princeton.edu/papers/wp0608f.pdf

Eficiencia de Mercados

El supuesto fundamental de la hipótesis de eficiencia consiste en que "los precios de los activos reflejan la totalidad de la información disponible". [Fama, 1970]. De acuerdo con Roberts (1976) se han definido clásicamente tres niveles de eficiencia según el conjunto de información que está reflejado en los precios de los activos: 1) eficiencia débil, que comprende toda la información histórica de precios, volúmenes, cotizaciones y demás información puramente transaccional; 2) La eficiencia semi-fuerte, que hace referencia a la información publica de las compañías, en cuanto a información financiera, económica y contable, información de la industria y del ambiente macroeconómico en general; y 3) eficiencia fuerte: cuando el conjunto de información comprende además de las anteriores, la información privada de los agentes con información privilegiada de la empresa ("insiders").

Un mercado es eficiente en su forma débil si la información histórica de precios, volúmenes y otras variables transaccionales no contiene información sobre el comportamiento de los precios futuros, o en cualquier caso no puede aprovecharse económicamente para tranzar en los activos. De acuerdo con lo anterior, en un mercado eficiente no es posible mediante el uso de métodos de Análisis Técnico, obtener rendimientos superiores a los alcanzados mediante una estrategia pasiva. En términos generales, las pruebas iniciales para medir el nivel de eficiencia débil del mercado identifican si los rendimientos pasados predicen en algún grado los rendimientos futuros [Campbell et al 1997; Maya y Torres 2004]. Alternativamente, una forma de medir la eficiencia débil es determinar el potencial de utilidades que se obtendrían mediante el uso de señales de compra y de venta generadas por un sistema operacional fundamentado en reglas técnicas [Elton y Gruber, 1984]

En resumen, "Si los movimientos en los precios de las acciones se pudieran prever, eso seria la prueba evidente de la ineficiencia del mercado de valores, porque la capacidad de predecir los precios indicaría que toda la información

disponible aun no había modificado el precio de las acciones. Por tanto, la idea de que las acciones ya reflejan toda la información disponible se denomina hipótesis del mercado eficiente." (bodie, kane y marcus 2004)

Gestión activa de carteras

La gestión activa de carteras aboga por hacer análisis mas profundos y por técnicas poco usuales, las cuales son las que generan más valor y de esta forma generan mayores beneficios.

Quienes están en pro de la hipótesis de eficiencia de los mercados prefieren una estrategia de inversión pasiva, la cual tiene como objetivo tener un portafolio de inversiones diversificado, sin intentar encontrar valores subvalorados o sobrevalorados.

"Debido a que la teoría del mercado eficiente indica que los precios de las acciones están a niveles justos, dada toda la información disponible, no tiene sentido comprar y vender valores frecuentemente, pues las transacciones generan grandes gastos de contratación sin incrementar el comportamiento esperado" (bodie, kane y marcus 2004).

En la práctica la configuración de carteras, involucra elementos adicionales a la construcción de rendimientos promedios esperados, que siguen una función de distribución difícil de estimar, y más bien, los inversionistas se apoyan en herramientas como el análisis fundamental que les permite hacer un seguimiento a variables macroeconómicas, sectoriales y empresariales. Adicionalmente se apoyan también en el análisis técnico para tratar de predecir el comportamiento futuro del precio de los activos (Sánchez et al, 2000). La utilización de estas dos herramientas apoya la conformación de portafolios en el sentido de que la predicción de los precios de los activos puede sustentar el proceso de decisión del inversionista en cuanto a la compra venta de activos.

Así pues, desde la creación de las bolsas de valores alrededor del mundo, los investigadores de diferentes áreas del conocimiento que van desde la estadística, la economía, las matemáticas, hasta la psicología, han diseñado teorías y herramientas para apoyar la toma de decisiones de inversión en el mercado público de valores. Dichos modelos intentan resolver preguntas fundamentales como qué activos adquirir, cuánto capital comprometer, cuándo invertir y cuándo salir de las posiciones.

A lo largo del tiempo los modelos planteados se han ido refinando y creciendo en complejidad. En particular, durante las últimas décadas, un campo que ha ganado bastante aceptación para el desarrollo de modelos de este tipo, ha sido el análisis Fundamental y Técnico. (Leigh et al. 2002). A continuación se describe las herramientas de Análisis Técnico y el Análisis Fundamental:

Análisis Fundamental

Un estudio de análisis fundamental examina los resultados macroeconómicos de un país, sus sectores, sus operaciones y la región en la cual está inmersa esta nación para entender el potencial de estabilidad y de crecimiento de dicho país. Los factores del país a considerar pueden incluir el crecimiento de la producción, el nivel de desempleo, el tamaño de sus mercados de valores, su estabilidad económica y política, la cantidad de deuda que posee, las proyecciones macroeconómicas, los niveles de riesgo, la estabilidad normativa, entre otros.

En la evaluación y análisis de estos estudios se utilizan diferentes razones, índices, u otros indicadores, para determinar el nivel de precio de su divisa o tasa de cambio al cual se alcanza estabilidad y fiabilidad de los flujos e indicadores macroeconómicos de dicha nación. Es de anotar, que este tipo de índices, muchas veces no son en última instancia una medida objetiva puesto que deben saberse interpretar. De esta forma los analistas fundamentales parten del supuesto de que los movimientos de los precios de las divisas tienen causas económicas o de otro tipo y que dichas causas pueden identificarse y

ponderarse antes de producir su efecto en el mercado. A partir de la valoración de todos estos factores, se estima el valor intrínseco de la divisa de cada nación, y se comparan con su valor de mercado para determinar su estado de sobre-valoración o sub-valoración. Por tanto, el grado de certeza de este método dependerá de la cantidad, calidad y oportunidad de la información que se tenga, así como de la profundidad con que se estudie. (Sánchez et al, 2000). Es por ello que esta, es una metodología o práctica de negociación de largo plazo, pues es necesario que el mercado realmente reconozca un error de valoración en el activo financiero, para proceder a corregirlo y así presentarse un movimiento fundamental en el precio.

Se ha visto que el comportamiento de los precios no siempre responde de igual manera a cada factor, o incluso que no depende de las condiciones económicas del país, sino que también influyen aspectos de diferente índole, como los fenómenos políticos, climáticos, sicológicos e incluso algunos aspectos irracionales. Basado en estos argumentos, la herramienta de análisis técnico sostiene que es imposible conocer todas las razones que afectan el precio de un activo financiero y por tanto no es necesario conocerlos para entender y participar del mercado. (Sánchez et al, 2000).

“El análisis fundamental se basa en el juego de la oferta y la demanda, que hace que los precios suban, bajen o queden igual. El enfoque fundamental examina todos los factores relevantes que afectan al precio de un mercado para determinar el valor intrínseco de este. El valor intrínseco es lo que los fundamentos indican como valor real de algo según la ley de la oferta y la demanda. Si este valor intrínseco esta por debajo del precio actual del mercado, quiere decir que el mercado esta sobrevalorado y debe venderse. Si el precio de mercado esta por debajo del valor intrínseco, entonces el mercado esta infravalorado y debe comprarse”. (Murphy, 2000)

“la hipótesis de eficiencia de mercados predice que la mayor parte e los análisis fundamentales añaden poco valor, puesto que si los analistas confían en las ganancias públicamente disponibles y en la información de la industria, no es probable que la evaluación del analista sobre las perspectivas de la empresa

sea significativamente mas apropiada que la de otro analista. Hay muchas empresas bien informadas y bien financiadas que realizan dichos estudios de mercado, y a la luz de dicha competencia seria difícil no obtener los datos que no estén disponibles también para los demás analistas. Solo los analistas con una visión única obtendrán recompensa." (bodie, kane y marcus 2004)

"El truco no esta en identificar las empresas que son buenas, sino en encontrar las que son mejores que lo que indican las estimaciones de los demás. Por esto no es suficiente con hacer un buen análisis de una compañía; solo se puede ganar dinero si su analista es mejor que el de sus competidores, porque se espera que el precio del mercado refleje ya toda la información disponible" (bodie, kane y marcus 2004)

Análisis Técnico

El análisis técnico tuvo sus orígenes dentro de los mercados accionarios, y posteriormente se extendió al mercado de futuros. Sin embargo, sus principios y herramientas son aplicables al estudio de las gráficas de cualquier instrumento financiero. El origen del análisis técnico tiene su fundamento en las teorías expuestas por Charles H. Dow a finales del siglo XIX. Se basa en una serie de premisas básicas sobre el funcionamiento de los mercados, empleando para ello los niveles de cierre de las gráficas diarias. Los trabajos de Dow se centraban en describir el comportamiento del mercado, sin adoptar una decisión de aprovechar la evolución futura de los valores. Su idea no era anticipar las tendencias, sino que buscó reconocer la aparición de mercados alcistas o bajistas.

La Teoría Dow

La teoría Dow, es una teoría basada en las escrituras recogidas del co-fundador del Dow Jones y el redactor Charles Dow, las cuales inspiraron el uso

y el desarrollo cada vez más extensos del análisis técnico desde finales del siglo 19. Desde entonces se han producido nuevas herramientas y teorías e incluso las herramientas ya existentes se han ido mejorando a grandes velocidades en las últimas décadas, con un espacial énfasis en las técnicas asistidas por computador. (Dong, Zhou 2002, Becker, Seshadri. 2003).

"Las seis premisas básicas de la Teoría de Dow son:

1. Los índices lo descuentan todo, teniendo en cuenta toda la información sobre los hechos que pueden afectar a los mercados.
2. Los mercados siguen tres tipos de tendencias o movimientos: *Una tendencia ascendente o descendente sigue un patrón de picos y valles cada vez más altos o más bajos dependiendo de la dirección de la tendencia.*
    1. Tendencias primarias o de largo plazo (seis meses a un año, o más): el mar
    2. Tendencias secundarias o de mediano plazo (tres semanas a tres meses): las olas
    3. Tendencias terciarias, menores o de corto plazo (menos de 3 semanas): las ondas
3. Las tendencias primarias siguen tres fases en su evolución:
    1. Fase de acumulación o de compra institucional.
    2. Fase fundamental o compra por parte del público en general.
    3. Fase de distribución, especulativa o de venta institucional.
4. Los diferentes índices bursátiles deben confirmar las tendencias alcistas o bajistas: varios índices deben confirmar las tendencias (financiero, industrial, etc.)
5. El volumen confirma la tendencia: el volumen de operación debe subir conforme el precio se mueve en la dirección la tendencia y bajar cuando el precio va en contra de ésta.
6. Una tendencia se mantiene vigente hasta el momento en que muestre señales claras de cambio de dirección."[2]

[2] Tomado de J.J Murphy Análisis Técnico de los Mercados Financieros

Desde el punto de vista del análisis técnico todos los movimientos de los precios estudiando sus gráficas reflejan el comportamiento a la alza o a la baja del mercado y de esta forma se identifican las tendencias sin la necesidad de un análisis fundamental

Para comprender la filosofía del Análisis Técnico y la operación de este tipo de estrategias es necesario comprender algunos aspectos de la dinámica de los mercados. En su forma elemental, el movimiento de los precios es el resultado de dos fuerzas opuestas: los compradores (demanda), y los vendedores (oferta). Cuando estas fuerzas no están en equilibrio, se dice que el mercado está en tendencia pues hay un movimiento sostenido del precio, (cuando es ascendente se conoce como tendencia alcista o "Bull", ó descendente conocido como tendencia bajista o "Bear"). Sin embargo, en períodos intermedios o al final de dichas tendencias, existen movimientos donde las fuerzas parecen estar llegando a un equilibrio originando un período conocido como de negociación o "Trading", en el cual los compradores y los vendedores intercambian el activo, pero éste no presenta una variación neta importante en su precio (Sánchez, Núñez y Couto; 2000).

Las series y gráficas son un reflejo de todos los factores que operan en un mercado. Los precios en el mercado descuentan inmediatamente cada pedazo de información nueva que pueda influenciarlo. Por tanto, no es necesario un análisis explícito sobre la situación económica, política, entre otros factores para predecir el comportamiento futuro de los precios. (Sánchez et al, 2000).

"El análisis técnico puede subdividirse en dos categorías:

1. Análisis gráfico o análisis chartista: El análisis gráfico utiliza líneas y figuras para identificar tendencias y patrones dentro del desempeño, principalmente de los precios, aunque también de los volúmenes e intereses compuestos (Rodríguez, Negrete y Santamaría; 2001). Supone que los precios a medida que suben y bajan, oscilan formando estructuras que pueden ser delimitadas con líneas, conformando figuras como triángulos, banderines,

rectángulos, rombos, entre muchos otros, los cuales representan períodos en los que se está consolidando una tendencia o por el contrario, en los que se está generando una reversa en ella (Sánchez, Núñez y Couto; 2000).

2. Análisis técnico en sentido estricto: La segunda forma del Análisis Técnico es el análisis cuantitativo, el cual utiliza herramientas estadísticas (indicadores) para ayudar a identificar períodos de sobre-compra o sobreventa del activo (Rodríguez, Negrete y Santamaría; 2001). Los indicadores técnicos son básicamente modelos matemáticos y estadísticos sobre los valores históricos de precios, volúmenes y volatilidades de los activos (Sánchez, Núñez y Couto; 2000); los cuales se calculan al final del período de análisis y que son llevados a gráficos para una mejor compresión (Elvira y Puig, 2000). Estos indicadores pretenden eliminar la subjetividad propia del análisis gráfico, así como la dificultad para identificar y sistematizar las señales a la hora de tomar las decisiones de inversión, proporcionando señales claras y precisas que no se prestan para controversias en cuanto a su interpretación, lo cual si sucede en el Chartismo (Murphy, 2000).

En el análisis técnico existen 3 tipos de indicadores gráficos, los seguidores de tendencia, los osciladores y los filtros.

Los indicadores seguidores de tendencia son indicadores que resultan útiles sólo en períodos en que los precios se encuentran en tendencias, ya que las identifican y confirman su comienzo, pero después de que se han consolidado. Su objetivo no es predecir el comportamiento futuro sino indicar su evolución, por lo cual los sistemas de operación basados en estos indicadores no logran comprar en el precio más bajo ni vender en el más alto, debido a que se pierde la primera parte del movimiento para poderlo confirmar (Medina, 2001). Cuando el mercado se encuentra en períodos de trading, los seguidores de tendencia son de muy poca utilidad. Entre los indicadores seguidores de tendencia se destacan principalmente los promedios móviles y los indicadores formados a partir de ellos (como el MACD y el TRIX), aunque existen otros que

se basan en premisas distintas, de los cuales el más conocido es el Sistema Parabólico.

De otro lado, los osciladores son otro tipo de herramientas opuestas en su utilidad a los seguidores de tendencia, ya que son muy útiles en mercados donde la tendencia no se ha consolidado, es decir, en aquellos en que los precios se mueven lateralmente, fluctuando entre bandas horizontales (períodos conocidos como trading) (Sánchez, Núñez y Couto; 2000), los cuales son característicos hacia el final de movimientos importantes (tendencias) (Murphy, 2000), mientras que en mercados con tendencia definida, son poco útiles. Estos osciladores también pueden detectar la pérdida de ímpetu en una tendencia antes de que ésta se haga evidente en los precios (Sánchez, Núñez y Couto; 2000). Entre los osciladores se destacan el Momento, la Tasa de Cambio (ROC), el Estocástico de Lane, el R% de Larry Williams, el Índice de Fuerza Relativa (RSI), algunos osciladores de volumen (como el ID, el Balance de Volumen Filtrado, el NVI, el PVI y el OVC) y osciladores de volatilidad como el RVI.

Los filtros son indicadores utilizados para determinar si el mercado se encuentra en tendencia ya sea a la alza o la baja o si se encuentra en movimiento horizontal, para permitir así escoger el uso de los 2 indicadores de tendencia anteriores, utilizando los seguidores de tendencia cuando existe una y los osciladores cuando el mercado se encuentra en movimiento horizontal.

IV. OBJETIVOS GENERAL Y ESPECIFICOS DEL PROYECTO

- **Objetivo general:**

  Automatizar el cálculo, la graficación y la optimización de algunos indicadores técnicos, aplicados a instrumentos de renta variable colombianos.

- **Objetivos específicos:**

  - Desarrollar el modulo encargado de obtener los precios de apertura, cierre, máximo, mínimo y volumen de las acciones
  - Desarrollar algoritmos para el cálculo de indicadores seleccionados.
  - Desarrollar algoritmos para la graficación de los indicadores técnicos calculados.
  - Para algunos de los indicadores seleccionados, desarrollar un algoritmo de optimización, para encontrar los parámetros óptimos, basados en un período y método de optimización.

V. ALCANCE

El análisis técnico parte de que los mercados no son eficientes en su forma débil, pues se pueden obtener beneficios económicos a partir de datos históricos tanto de precios como de otras variables transaccionales como el volumen.

Esta investigación sólo tomará algunas de las acciones del mercado de valores colombiano, así como también tomará sólo algunos indicadores técnicos. Y a partir de allí, desarrollar un software que automatiza el cálculo, la graficación y la optimización de reglas técnicas, y así servir como herramienta para el análisis técnico del mercado de valores colombiano

VI. METODOLOGÍA

Sistema desarrollado en el lenguaje de programación Visual C# .NET, con bases de datos en SQL y modulo de adquisición de precios en Visual Basic para Aplicaciones.

El lenguaje de programación Visual C# tiene las herramientas necesarias para el diseño y desarrollo de una interfaz grafica amigable y versátil para mostrar los resultados de los análisis técnicos, por otro lado para la adquisición de la información de los precios de los activos se hace muy útil el lenguaje de programación Visual Basic para Aplicaciones utilizado para obtener la información de los precios diariamente desde Internet y organizarlos por empresas inscritas en la bolsa de valores de Colombia en formatos de Excel.

A la hora de leer la información desde el formato Excel con Visual C#, se utilizan las herramientas de Microsoft Visual Studio tools for the Microsoft Office System, que permiten tomar objetos Excel y procesarlos en aplicaciones .NET.

Pero este flujo de información desde formatos Excel a Aplicaciones Visual C# es lento, por lo cual se hace necesario utilizar Bases de Datos en SQL para guardar toda la información histórica y poder acceder a ella de una forma rápida para hacer los cálculos y las graficas necesarias.

Para los cálculos y las graficas del Análisis experimental se usan las herramientas teóricas publicadas por los diferentes autores como JJ-MURPHY y la implementación de estas por medio de la programación del sistema, desarrollando los algoritmos.

- **Características del software:**

    - El software debe bajar automáticamente la información diaria de los precios que son valor de apertura, cierre, máximo, mínimo y volumen de las empresas pertenecientes al IGBC en la bolsa de valores.

    - Mencionado anteriormente el software debe graficar los precios y las diferentes graficas de indicadores estadísticos y velas japonesas

- **Beneficiarios**

  Este software servirá como herramienta complementaria al análisis fundamental de los mercados de valores que se hace por parte de los inversionistas.

- **Presupuesto:**

  Para los estudios del mercado y programación del software es necesario un computador Pentium D con 2GB de RAM y acceso a Internet de banda ancha.

- **Observaciones**

  El software resultante de este proyecto de grado será una herramienta para la ayuda en el análisis de los mercados de valores, mas no un sistema para toma de decisiones automáticas.

VII. Trabajo Futuro:

- Desarrollo de módulos avanzados con sistemas expertos para la toma de decisiones de compra y venta.
- Desarrollo de algoritmos de identificación de patrones de cabeza y hombros, ondas elliot, líneas de abanico de Gann y Fibonacci y modelos de continuidad.
- Desarrollo de algoritmos para interpretación de variables macroeconómicas, tipos de cambio e indicadores de bolsas internacionales.
- Implantación de un modulo para empalme con un sistema Data IFX para análisis en tiempo real.
- Desarrollo de módulos de optimización de Oscilador Estocástico y Larry Williams.

CAPITULO 2: MODULOS GRAFICOS Y DE OBTENCION DE PRECIOS

- **Modulo de obtención de precios desde Excel**

Debido a que la información fuente para nuestros análisis casi siempre está en formatos de Excel, se hace fácil el almacenamiento en libros. Ya que el flujo de información desde Excel a una aplicación en Visual C# es significativamente lento al manejar cantidades grandes de información, se hace necesario la implementación de una base de datos en SQL para pasar la información de precios de las acciones desde Excel a esta, antes de hacer los análisis técnicos.

Este modulo consiste en obtener la información desde Excel y después de almacenarla en la base de datos mostrarla en pantalla. El usuario al tener la información en pantalla puede modificarla y los cambios correspondientes se almacenaran en la base de datos. Además este modulo puede borrar la información de la acción para obtenerla de nuevo desde Excel. (Ver figura 1)

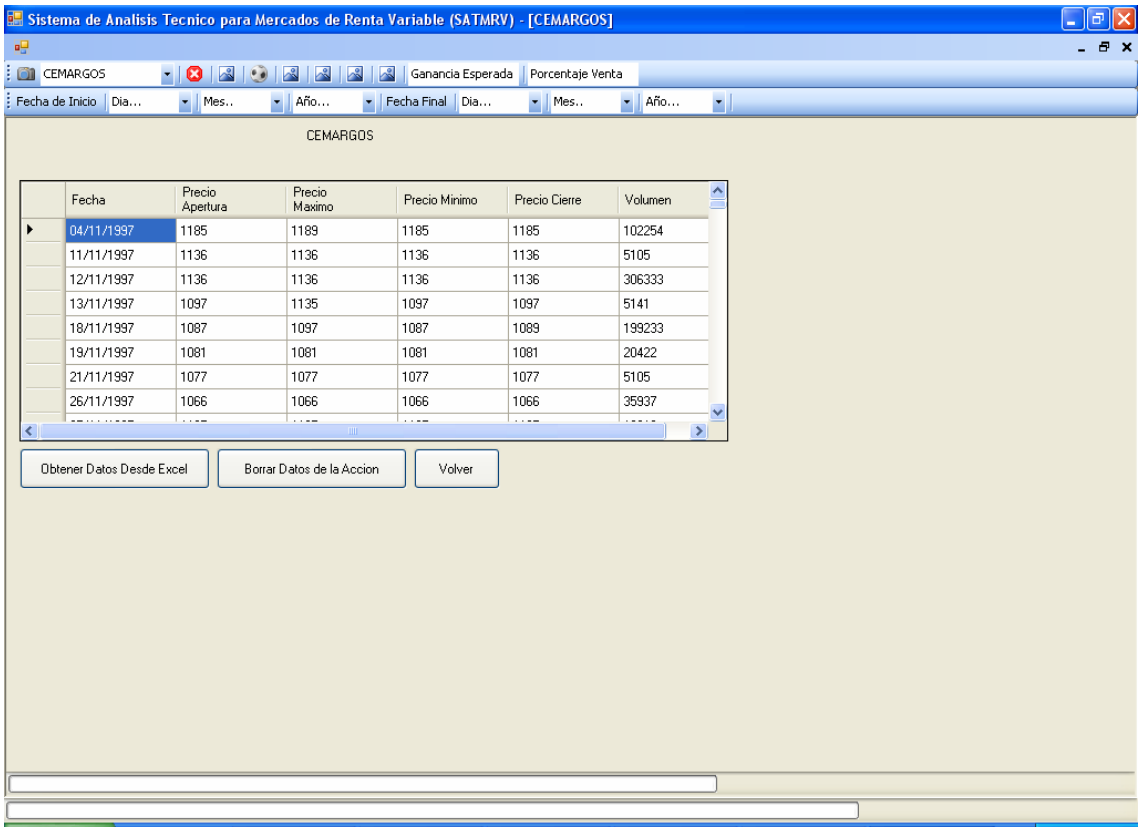

FIGURA 1

- **Modulo grafico de precios de cierre**

Este modulo se encarga de tomar la información de los precios de la acción y graficar sus precios de cierre y dos medias móviles con sus períodos correspondientes. La información diaria de los precios se puede ver en la barra de herramientas a medida que el Mouse se desplaza por la grafica, también se puede acercar y alejar la imagen para su mejor visualización. (Ver figura 2).

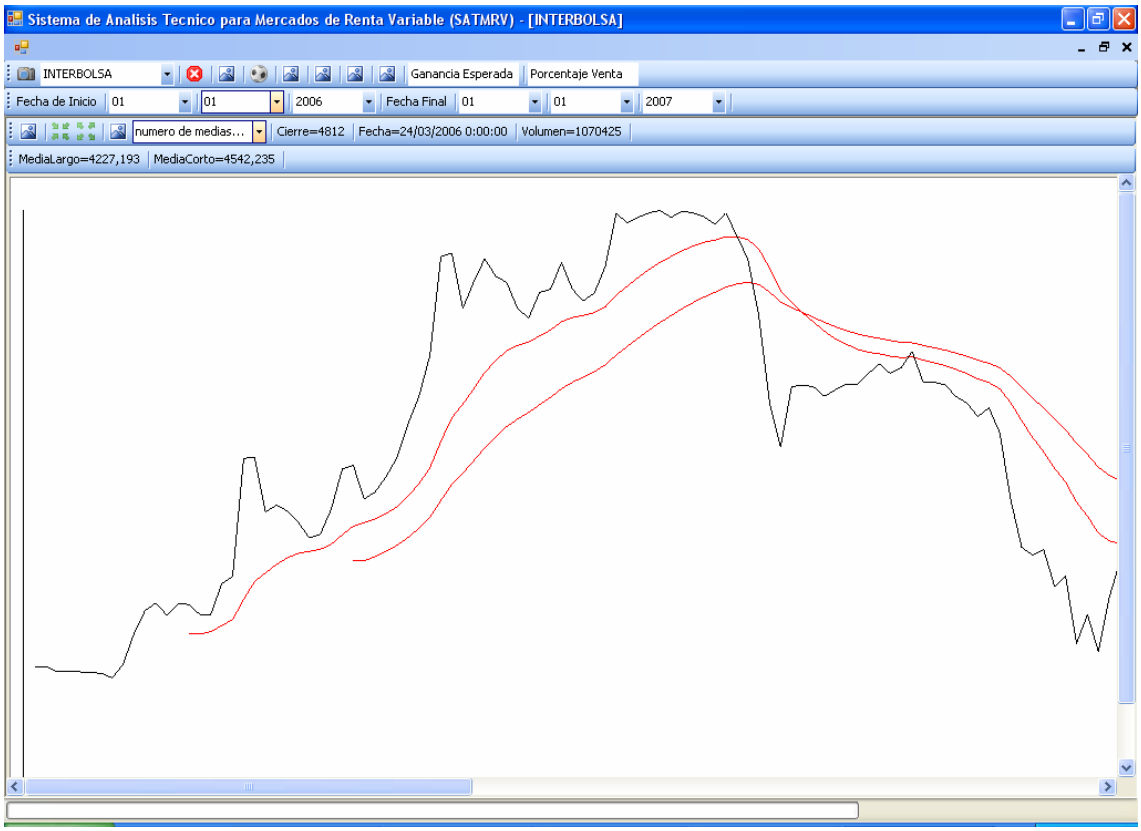

FIGURA 2: La curva de color negro es la de los precios de cierre y las rojas son las medias móviles de corto y largo plazo

- **Módulo gráfico de velas japonesas**

Muy similar al anterior, este módulo grafica los precios máximo, mínimo apertura y cierre en forma de velas japonesas, tiene las mismas funciones que el módulo anterior, con la diferencia de que aquí se muestran todos los precios del día mientras en el anterior solo los de cierre, y para las velas japonesas sólo necesitamos una sola media móvil exponencial. (Ver figura 3)

FIGURA 3

- **Módulo gráfico de volumen**

Similar a los módulos anteriores, este grafica los volúmenes con barras azules que comienzan desde abajo, y con la posibilidad también de dibujar sólo una media móvil exponencial (Ver figura 4).

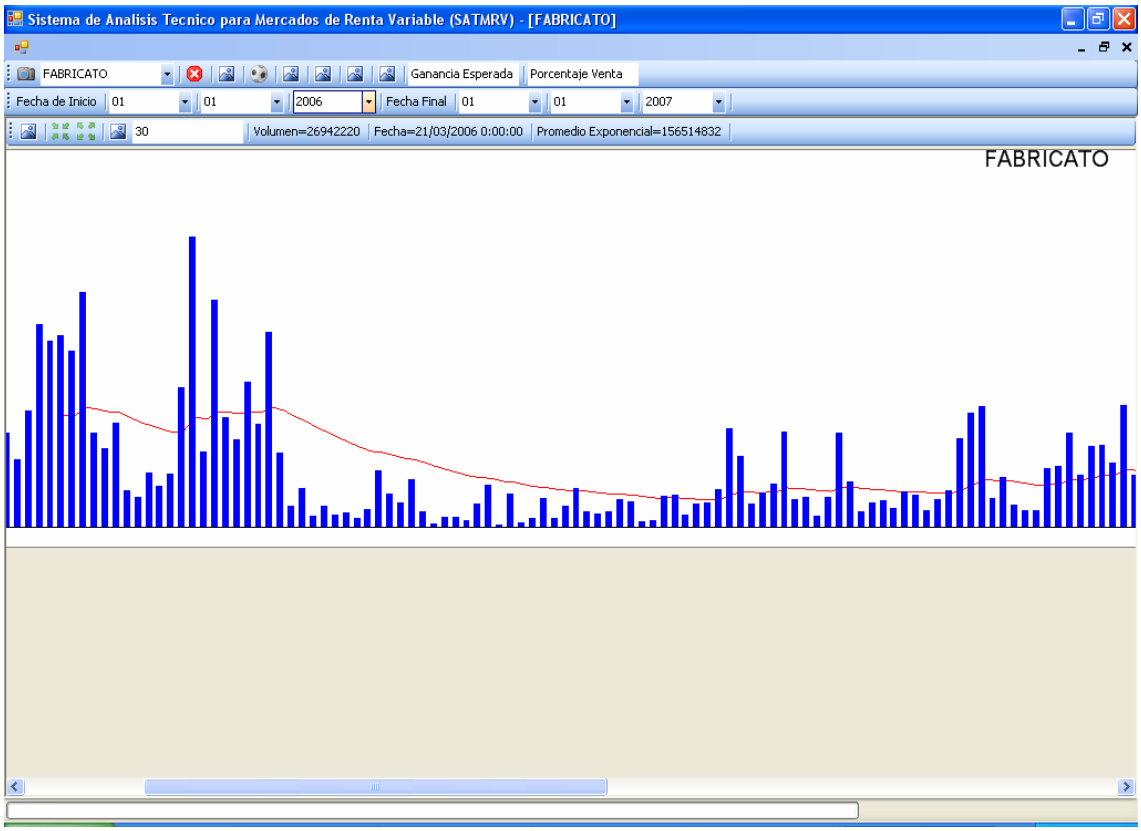

FIGURA 4

CAPITULO 3: HALLAZGO DE PATRONES DE REVERSA DE VELAS JAPONESAS

“Aunque los japoneses han estado utilizando esta técnica de gráficos y análisis desde hace siglos, en el mundo occidental se ha hecho conocida en años recientes.” Murphy 2000

Entre los muchos patrones de velas japonesas, en el presente proyecto, se toman los patrones de reversa alcista y bajista sugeridos en el paper Candlestick technical trading strategies: Can they create value for investors? Ben R. Marshall, Martin R. Young, Lawrence C. Rose.

Los patrones de reversa de velas japonesas, son los formados con velas de 2 o más días, estos indican un posible cambio de dirección en la tendencia de los precios.

Los patrones con los cuales trabajamos en este proyecto son los siguientes:

Patrones de reversa alcista: indican un cambio de tendencia de bajista a alcista

- Hammer
- Bullish Engulfing
- Piercing Line
- Bullish Harami
- Three Inside Up
- Three Outside Up
- Tweezer Bottom

Patrones de reversa bajista: indican un cambio de tendencia de alcista a bajista

- Hanging Man
- Bearish Engulfing
- Dark Cloud Cover
- Bearish Harami
- Three Inside Down

- Three Outside Down
- Tweezer Top

Para encontrar los patrones mencionados, se desarrolló un algoritmo para identificar las diferentes fechas en las que se cumplieron dichos patrones en las acciones cotizadas en la Bolsa de Valores de Colombia, desde la fusión de 2001, el cual identifica cada nuevo patrón que se va formando día a día.

En la interfaz gráfica encargada de mostrar los resultados de los análisis de cada acción se pueden ver las fechas en las cuales se cumplen los patrones de reversa alcistas y bajistas (ver en la Figura 4 y 5, respectivamente).

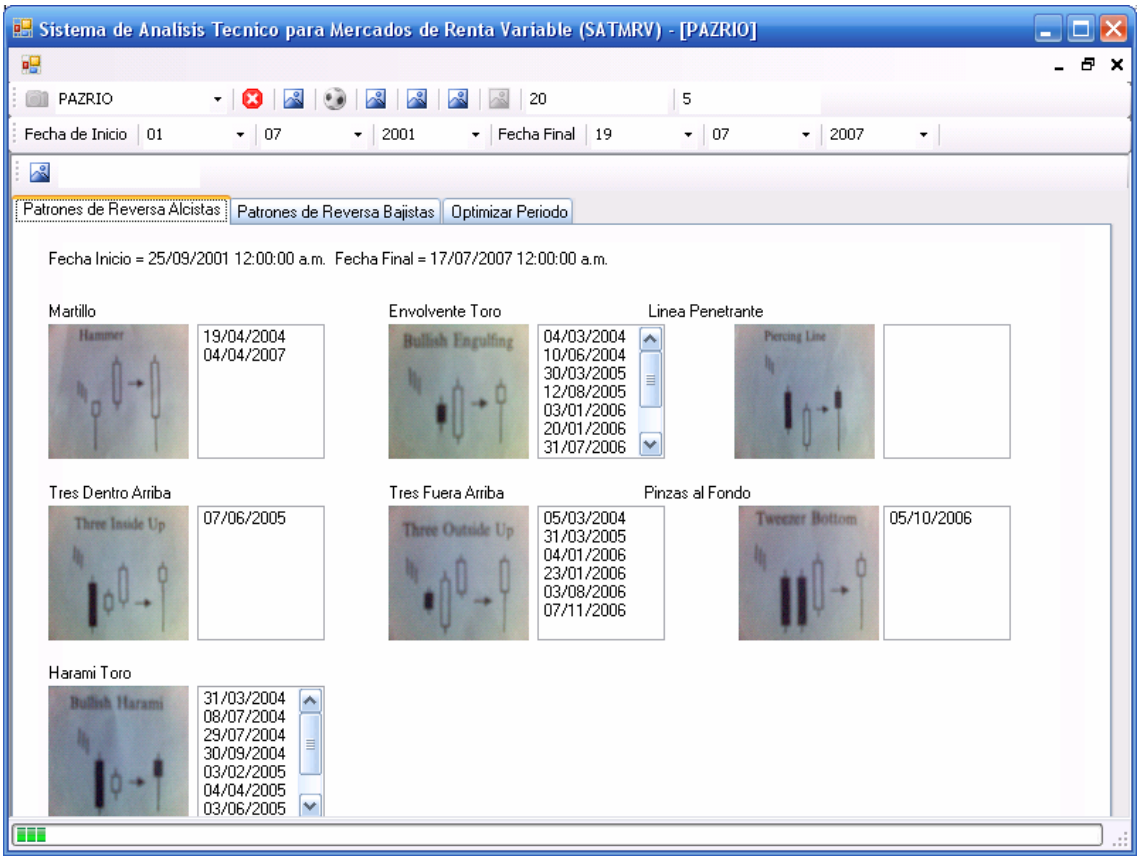

FIGURA 4.

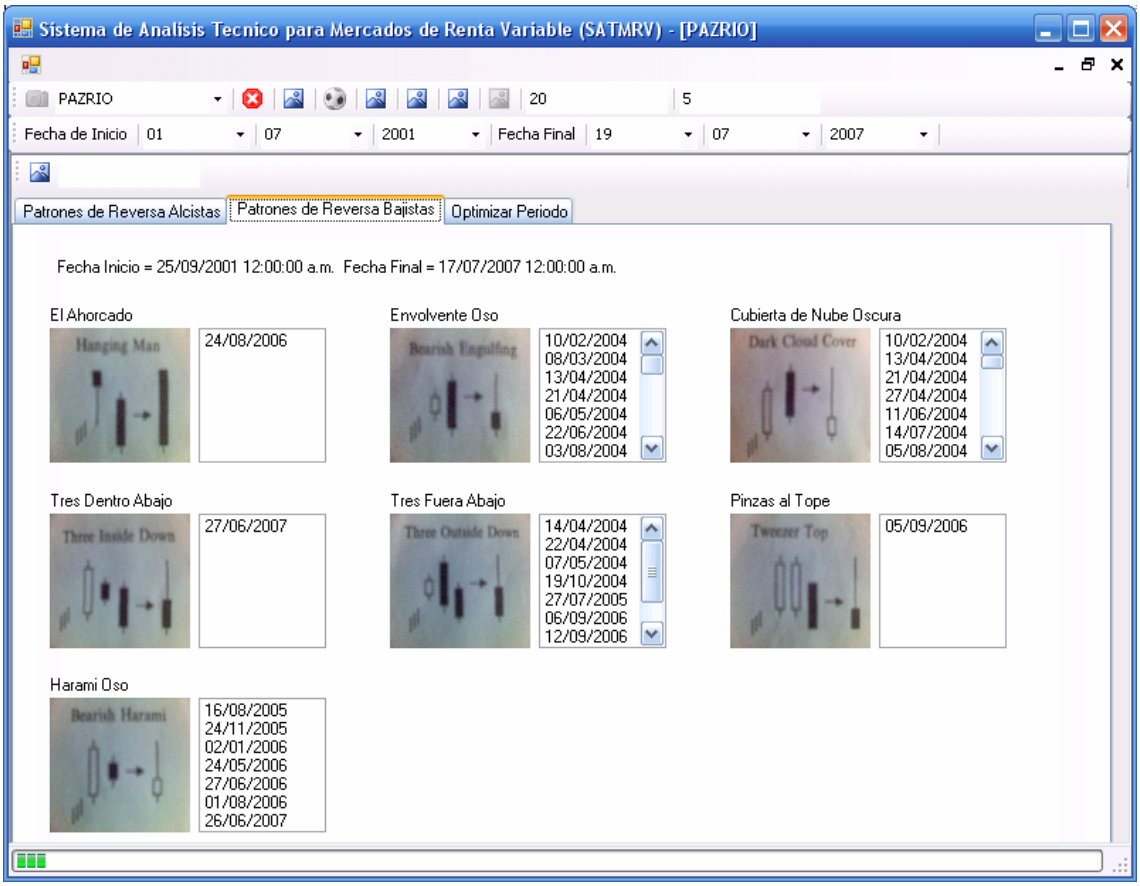

FIGURA 5.

El siguiente es el código fuente desarrollado en el lenguaje de programación Visual C# de la función para hallar los patrones de velas japonesas utilizando el algoritmo desarrollado.

```
//Función para hallar los patrones de velas japonesas en el rango
especificado y clasificarlas en las listas
private void hallarPatronesDeVelasJaponesas(ArrayList listaPrecios)
{
//////////////////////////////////////////////////////////////////////
////Patrones Bearish de Reversion

      PantallaPrincipal.dia primerDia;
      PantallaPrincipal.dia segundoDia;
      PantallaPrincipal.dia tercerDia;

//Listas que tienen todos los patrones alcistas y bajistas para la
//optimización
      listaPatronesReversaAlcistas = new ArrayList();
      listaPatronesReversaBajistas = new ArrayList();

//Por cada día en la lista de precios se verifica si cumple un patrón
//y se clasifica en la lista correcta, además clasifica todos los
//alcistas y los bajistas en otras dos listas diferentes
      for (int i = 0; i < listaPrecios.Count - 1; i++)
      {
          primerDia = (PantallaPrincipal.dia)listaPrecios[i];
          segundoDia = (PantallaPrincipal.dia)listaPrecios[i + 1];
```

```
//Si es el penúltimo día de la lista de precios, no se pueden obtener
//patrones compuestos de tres días, por eso si esto es así se hace el
//valor del tercer día igual a cero para poder hallar los patrones de
//2 días en el día que se hace el análisis
       try
       {
            tercerDia = (PantallaPrincipal.dia)listaPrecios[i + 2];
       }
       catch {
            tercerDia.valorCierre = 0;
            tercerDia.valorApertura = 0;
            tercerDia.fecha = segundoDia.fecha;
       }

//verifica si la fecha cumple el patrón bullish engulfing
//Verifica que el precio de cierre sea menor que el precio de apertura
      if (primerDia.valorApertura > primerDia.valorCierre)
       {
//verifica que el precio de apertura del día siguiente sea menor que
//el precio de cierre del día anterior y que el cierre del día
//siguiente sea mayor que el precio de apertura del día anterior
           if (segundoDia.valorApertura < primerDia.valorCierre &&
            segundoDia.valorCierre > primerDia.valorApertura)
           {
//verifica que el precio de cierre y el de apertura de los días no sea
//igual
                if (primerDia.valorApertura != primerDia.valorCierre
                 && segundoDia.valorApertura !=
                 segundoDia.valorCierre)
                {
                lbEnvolventeToro.Items.Add(segundoDia.fecha);
                listaPatronesReversaAlcistas.Add(segundoDia.fecha);
                }
           }
      }

//verifica si la fecha cumple el patrón hammer
//Verifica si el precio de cierre es mayor que el de apertura
      if (primerDia.valorApertura < primerDia.valorCierre)
      {
//Verifica que la sombra inferior sea por lo menos 2 veces mas grande
//que la barra
           if ((primerDia.valorCierre - primerDia.valorApertura) <
           (primerDia.valorApertura - primerDia.valorMinimo) / 2)
           {
//Verifica que la sombra superior sea muy poca
                if ((primerDia.valorCierre -
primerDia.valorApertura)
                / 2 > (primerDia.valorMaximo -
                primerDia.valorCierre))
                {
//verifica que el precio de apertura del día siguiente sea menor que
//su precio de cierre
                     if (segundoDia.valorApertura <
                          segundoDia.valorCierre)
                     {
//verifica que el precio de apertura del día siguiente sea mayor o
//igual al precio de cierre del día anterior
                          if (segundoDia.valorApertura >=
                               primerDia.valorCierre)
```

```
                                {
//verifica que el precio de cierre y de apertura de los días no sea
//igual
                                    if (primerDia.valorApertura !=
                                        primerDia.valorCierre &&
                                        segundoDia.valorApertura !=
                                        segundoDia.valorCierre)
                                {
                                lbMartillo.Items.Add(segundoDia.fecha);
                    listaPatronesReversaAlcistas.Add(segundoDia.fecha);
                                }
                                }
                            }
                        }
                    }
            }

//verifica si la fecha cumple el patrón Piercing line
//Verifica si el precio de cierre es menor que el de apertura
            if (primerDia.valorApertura > primerDia.valorCierre)
            {
//Verifica que el precio de apertura del día siguiente sea menor que
//el precio de cierre del día anterior
                    if (segundoDia.valorApertura <
                    primerDia.valorCierre)
                    {




//Verifica que el precio de cierre del día siguiente esté por lo menos
//más arriba de la mitad de la suma de los precios apertura y de
//cierre del día anterior, y menor que el precio de apertura del
//primer día
                            if (segundoDia.valorCierre >
                            (primerDia.valorApertura +
                            primerDia.valorCierre) / 2 &&
                            segundoDia.valorCierre <
                            primerDia.valorApertura)
                             {
//verifica que el precio de cierre y de apertura de los días no sea
//igual
                                    if (primerDia.valorApertura !=
                                    primerDia.valorCierre &&
                                    segundoDia.valorApertura !=
                                    segundoDia.valorCierre)
                            {
                            lbLineaPenetrante.Items.Add(segundoDia.fecha);
                    listaPatronesReversaAlcistas.Add(segundoDia.fecha);


                            }
                        }
                    }
            }

//verifica si la fecha cumple el patrón bullish harami
//Verifica que el precio de cierre sea menor que el precio de apertura
```

```
        if (primerDia.valorApertura > primerDia.valorCierre)
        {
//verifica que el precio de apertura del día siguiente sea mayor que
//el precio de cierre del día anterior y que el cierre del día
//siguiente sea menor que el precio de apertura del día anterior y que
//el cierre del día siguiente sea mayor que su apertura.
            if (segundoDia.valorApertura > primerDia.valorCierre &&
            segundoDia.valorCierre < primerDia.valorApertura &&
            segundoDia.valorCierre > segundoDia.valorApertura)
            {
//verifica que el precio de cierre y de apertura de los días no sea
igual
                if (primerDia.valorApertura != primerDia.valorCierre
                && segundoDia.valorApertura !=
                segundoDia.valorCierre)
                {
                            lbHaramiToro.Items.Add(segundoDia.fecha);
                listaPatronesReversaAlcistas.Add(segundoDia.fecha);

                }
             }
        }
//Trata de hallar los patrones de tres días, si es el penúltimo día,
//no se puede, pero se utiliza el try para que no lance excepción
        try{
//verifica si la fecha cumple el patrón Three Inside Up
//Verifica que el precio de cierre sea menor que el precio de apertura
            if (primerDia.valorApertura > primerDia.valorCierre)
            {
//verifica que el precio de apertura del día siguiente sea mayor que
//el precio de cierre del día anterior, que el cierre del día
//siguiente sea menor que el precio de apertura del día anterior y que
//el cierre del día siguiente sea mayor que su apertura.
                if (segundoDia.valorApertura > primerDia.valorCierre
                && segundoDia.valorCierre < primerDia.valorApertura
                && segundoDia.valorCierre > segundoDia.valorApertura)
                {
//verifica que el precio de apertura del tercer día sea mayor que el
//precio de apertura del segundo y que su cierre sea mayor que el
//precio de apertura del primer día
                    if (tercerDia.valorApertura >
                    segundoDia.valorCierre && tercerDia.valorCierre
                    > primerDia.valorApertura)
                    {
//verifica que el precio de cierre y de apertura de los días no sea
//igual
                        if (primerDia.valorApertura !=
                        primerDia.valorCierre &&
                        segundoDia.valorApertura !=
                        segundoDia.valorCierre &&
                        tercerDia.valorApertura !=
                        tercerDia.valorCierre)
                        {
                    lbTresDentroArriba.Items.Add(tercerDia.fecha);
                listaPatronesReversaAlcistas.Add(tercerDia.fecha);


                        }
                    }
                }
            }
```

```
//verifica si la fecha cumple el patrón three outside up
//Verifica que el precio de cierre sea menor que el precio de apertura
             if (primerDia.valorApertura > primerDia.valorCierre)
             {
//verifica que el precio de apertura del día siguiente sea menor que
//el precio de cierre del día anterior y que el cierre del día
//siguiente sea mayor que el precio de apertura del día anterior
                 if (segundoDia.valorApertura < primerDia.valorCierre
                 && segundoDia.valorCierre > primerDia.valorApertura)
                 {
//verifica que el precio de apertura del tercer día sea mayor que el
//precio medio del segundo día y que su cierre sea mayor que el cierre
//del segundo día
                     if (tercerDia.valorApertura >
                     (segundoDia.valorApertura +
                     segundoDia.valorCierre) / 2 &&
                     tercerDia.valorCierre > segundoDia.valorCierre)
                     {
//verifica que el precio de cierre y de apertura de los días no sea
//igual
                         if (primerDia.valorApertura !=
                         primerDia.valorCierre &&
                         segundoDia.valorApertura !=
                         segundoDia.valorCierre &&
                         tercerDia.valorApertura !=
                         tercerDia.valorCierre)
                         {
                     lbTresFueraArriba.Items.Add(tercerDia.fecha);
                 listaPatronesReversaAlcistas.Add(tercerDia.fecha);
                         }
                     }
                 }
             }

//verifica si la fecha cumple el patrón Tweezer Bottom
//Verifica que el precio de cierre sea menor que el precio de apertura
             if (primerDia.valorCierre < segundoDia.valorCierre * 1.01
             && primerDia.valorCierre > segundoDia.valorCierre * 0.99 &&
             primerDia.valorApertura < segundoDia.valorApertura * 1.01
             && primerDia.valorApertura > segundoDia.valorApertura *
             0.99 && primerDia.valorApertura > primerDia.valorCierre)
             {
//verifica que el precio de apertura del día siguiente sea mayor que
//el precio de su cierre
                 if (segundoDia.valorApertura >
                 segundoDia.valorCierre)
                 {
//verifica que el precio de apertura del tercer día sea mayor que el
//precio de cierre del segundo día y que su cierre sea mayor que su
//apertura y también que el sel día anterior
                     if (tercerDia.valorApertura >
                     segundoDia.valorCierre && tercerDia.valorCierre
                     > tercerDia.valorApertura &&
                     tercerDia.valorCierre >
                     segundoDia.valorApertura)
                     {
//verifica que el precio de cierre y de apertura de los días no sea
//igual
                         if (primerDia.valorApertura !=
                         primerDia.valorCierre &&
```

```
                                    segundoDia.valorApertura !=
                                    segundoDia.valorCierre &&
                                    tercerDia.valorApertura !=
                                    tercerDia.valorCierre)
                              {
                        lbPinzasAlFondo.Items.Add(tercerDia.fecha);
                        listaPatronesReversaAlcistas.Add(tercerDia.fecha);
                              }
                        }
                  }
            }
}catch{}

////////////////////////////////////////////////////////////////////
//Patrones bearish de reversión

//verifica si la fecha cumple el patrón Hanging Man
//Verifica que el precio de cierre sea menor que el precio de apertura
if (primerDia.valorApertura > primerDia.valorCierre)
{
//Verifica que la sombra inferior sea por lo menos 2 veces más grande
//que la barra
      if ((primerDia.valorApertura - primerDia.valorCierre) <
      (primerDia.valorCierre - primerDia.valorMinimo) / 2)
      {
//Verifica que la sombra superior sea muy poca
            if ((primerDia.valorApertura - primerDia.valorCierre) / 2
            > (primerDia.valorMaximo - primerDia.valorApertura))
            {
//Verifica que en el segundo día el precio de cierre sea menor que el
//precio de apertura y que el precio de apertura sea menor que el de
//cierre del día anterior
                  if (segundoDia.valorApertura >
                  segundoDia.valorCierre
                  && segundoDia.valorApertura < primerDia.valorCierre)
                  {
//verifica que el precio de cierre y de apertura de los días no sea
//igual

                        if (primerDia.valorApertura !=
                        primerDia.valorCierre &&
                        segundoDia.valorApertura !=
                        segundoDia.valorCierre)
                        {

                              lbElAhorcado.Items.Add(segundoDia.fecha);
                  listaPatronesReversaBajistas.Add(segundoDia.fecha);

                        }
                  }
            }
      }
}

//verifica si la fecha cumple el patrón bearish engulfing
//Verifica que el precio de cierre sea mayor que el precio de apertura
if (primerDia.valorApertura < primerDia.valorCierre)
{
//verifica que el precio de apertura del día siguiente sea mayor que
//el precio de apertura del día anterior y que el cierre del día
//siguiente sea menor que el precio de apertura del día anterior
```

```
      if (segundoDia.valorApertura > primerDia.valorApertura &&
      segundoDia.valorCierre < primerDia.valorApertura)
      {
//verifica que el precio de cierre y de apertura de los días no sea
//igual
            if (primerDia.valorApertura != primerDia.valorCierre &&
            segundoDia.valorApertura != segundoDia.valorCierre)
            {
                  lbEnvolventeOso.Items.Add(segundoDia.fecha);
                listaPatronesReversaBajistas.Add(segundoDia.fecha);

            }
      }
}

//verifica si la fecha cumple el patrón Dark Cloud Cover
//Verifica que el precio de cierre sea mayor que el precio de apertura
if (primerDia.valorApertura < primerDia.valorCierre)
{
//verifica que el precio de apertura del día siguiente sea mayor que
//el precio de cierre del día anterior y que el cierre del día
//siguiente sea menor que el precio de cierre del día anterior
      if (segundoDia.valorApertura > primerDia.valorCierre &&
      segundoDia.valorCierre < primerDia.valorCierre)
      {
//verifica que el precio de cierre y de apertura de los días no sea
//igual
            if (primerDia.valorApertura != primerDia.valorCierre &&
            segundoDia.valorApertura != segundoDia.valorCierre)
            {
            lbCubiertaDeNuveOscura.Items.Add(segundoDia.fecha);
            listaPatronesReversaBajistas.Add(segundoDia.fecha);
            }
      }
}

//verifica si la fecha cumple el patrón Bearish Harami
//Verifica que el precio de cierre sea mayor que el precio de apertura
if (primerDia.valorApertura < primerDia.valorCierre)
{
//verifica que el precio de apertura del día siguiente sea menor que
//el precio de cierre del día anterior, que el cierre del día
//siguiente sea mayor que el precio de apertura del día anterior y que
//sea menor que el de apertura del segundo día
      if (segundoDia.valorApertura < primerDia.valorCierre &&
      segundoDia.valorCierre > primerDia.valorApertura &&
      segundoDia.valorCierre < segundoDia.valorApertura)
      {
//verifica que el precio de cierre y de apertura de los días no sea
//igual
            if (primerDia.valorApertura != primerDia.valorCierre &&
            segundoDia.valorApertura != segundoDia.valorCierre)
            {
            lbHaramiOso.Items.Add(segundoDia.fecha);
            listaPatronesReversaBajistas.Add(segundoDia.fecha);
            }
      }
}

//Trata de hallar los patrones de tres días, si es el penúltimo día
//no se puede, pero se utiliza el try para que no lance excepción
```

```
try{
//verifica si la fecha cumple el patrón Three Inside Down
//Verifica que el precio de cierre sea mayor que el precio de apertura
      if (primerDia.valorApertura < primerDia.valorCierre)
      {
//verifica que el precio de apertura del día siguiente sea menor que
//el precio de cierre del día anterior, que el cierre del día
//siguiente sea mayor que el precio de apertura del día anterior y que
//sea menor que el de apertura del segundo día
            if (segundoDia.valorApertura < primerDia.valorCierre &&
            segundoDia.valorCierre > primerDia.valorApertura &&
            segundoDia.valorCierre < segundoDia.valorApertura)
            {
//verifica que el precio de apertura del tercer día sea menor que el
//de apertura del segundo y que el precio de cierre del tercer día sea
//menor que el de apertura del primero
                  if (tercerDia.valorApertura <
                  segundoDia.valorApertura && tercerDia.valorCierre <
                  primerDia.valorApertura)
                  {
//verifica que el precio de cierre y de apertura de los días no sea
//igual
                        if (primerDia.valorApertura !=
                        primerDia.valorCierre &&
                        segundoDia.valorApertura !=
                        segundoDia.valorCierre &&
                        tercerDia.valorApertura !=
                        tercerDia.valorCierre)
                        {
                  lbTresDentroAbajo.Items.Add(tercerDia.fecha);
                  listaPatronesReversaBajistas.Add(tercerDia.fecha);
                  }
            }
     }
}

//verifica si la fecha cumple el patrón Three Outside Down
//Verifica que el precio de cierre sea mayor que el precio de apertura
if (primerDia.valorApertura < primerDia.valorCierre)
{
//verifica que el precio de apertura del día siguiente sea mayor que
//el precio de apertura del día anterior y que el cierre del día
//siguiente sea menor que el precio de apertura del día anterior
      if (segundoDia.valorApertura > primerDia.valorApertura &&
      segundoDia.valorCierre < primerDia.valorApertura)
      {
//verifica que el precio de apertura del tercer día sea menor que el
//cierre del primero, y que el cierre del tercer día sea menor que el
//cierre del segundo
            if (tercerDia.valorApertura < primerDia.valorCierre &&
            tercerDia.valorCierre < segundoDia.valorCierre)
            {
//verifica que el precio de cierre y de apertura de los días no sea
//igual
                  if (primerDia.valorApertura != primerDia.valorCierre
                  && segundoDia.valorApertura != segundoDia.valorCierre
                  && tercerDia.valorApertura != tercerDia.valorCierre)
                  {
                  lbTresFueraAbajo.Items.Add(tercerDia.fecha);
                  listaPatronesReversaBajistas.Add(tercerDia.fecha);
                  }
```

```
            }
      }
}

//verifica si la fecha cumple el patrón Tweezer Top
//Verifica que el precio de cierre tenga poca sombra con respecto al
//precio de cierre del día siguiente, que el precio de apertura sea
//igual al precio de apertura del día siguiente y que el precio de
//apertura del preimer día sea menor que el de cierre con márgen de 1%
if (primerDia.valorCierre < segundoDia.valorCierre * 1.01 &&
primerDia.valorCierre > segundoDia.valorCierre * 0.99 &&
primerDia.valorApertura < segundoDia.valorApertura * 1.01 &&
primerDia.valorApertura > segundoDia.valorApertura * 0.99 &&
primerDia.valorApertura < primerDia.valorCierre)
{
//Verifica que el precio de apertura del segundo día sea menor que el
//precio de cierre
      if (segundoDia.valorApertura < segundoDia.valorCierre)
      {
//verifica que el precio de apertura del tercer día sea menor que el
//precio de cierre del segundo día y que el cierre del tercer día sea
//menor que el precio de apertura del día anterior
            if (tercerDia.valorApertura < segundoDia.valorCierre &&
            tercerDia.valorCierre < segundoDia.valorApertura)
            {
//verifica que el precio de cierre y de apertura de los días no sea
//igual
                  if (primerDia.valorApertura != primerDia.valorCierre
                  && segundoDia.valorApertura != segundoDia.valorCierre
                  && tercerDia.valorApertura != tercerDia.valorCierre)
                  {
                  lbPinzasAlTope.Items.Add(tercerDia.fecha);
                  listaPatronesReversaBajistas.Add(tercerDia.fecha);
                  }
             }
          }
      }
  }
  catch { }
  }
}
```

CAPITULO 4: OPTIMIZACION DE PERIODO DE MEDIA MOVIL EXPONENCIAL COMO FILTRO DE PATRONES DE REVERSA DE VELAS JAPONESAS

"La media móvil es uno de los indicadores técnicos mas versátil y cuyo uso esta más extendido. Por la forma en que se hace y por el hecho de que puede cuantificarse y verificarse tan fácilmente, es la base de muchos sistemas mecánicos de seguimiento de tendencias en uso hoy en día." Murphy 2000

Una media móvil consiste en sacar el promedio de varios precios de cierre históricos para suavizar la grafica de estos con una curva. Esto nos permite hacer un filtro de los patrones de velas japonesas identificados.

Un patrón de reversa alcista, no tiene sentido si se presenta en una tendencia alcista, así como un patrón de reversa bajista no tiene sentido si se presenta en una tendencia bajista. Lo que se hace con la media móvil, es filtrar estos patrones que no tienen sentido para hallar los patrones adecuados para invertir.

La media móvil exponencial se halla con la siguiente formula:

$$ME(t) = ME(t-1) + F * (P - ME(t-1))$$

$$F=2 / (N+1)$$

Donde t es el dia, P es el precio de cierre del dia y N es el período de la media móvil.

Cuando el precio de cierre de un día corta en sentido ascendente el promedio de la media móvil, se considera que hay una tendencia alcista, y cuando la corta en sentido descendente se considera que hay una tendencia bajista.

Cuando el periodo de la media móvil es mas grande, la media móvil es menos sensible a los cambios en los precios en el corto plazo. Para hallar el período

correcto de la media móvil, se hacen iteraciones hallando la máxima ganancia que se hubiera obtenido con cada período desde 1 hasta 100.

En la figura 6 se puede ver los resultados del análisis de patrones de velas japonesas con filtro de media móvil exponencial, que son:

- El período óptimo

El mejor período de la media móvil encontrado por la ganacia histórica

- La ganancia bruta

Es la suma de las ganacias de cada transacción (Que haya dado como resultado ganacia)

- La pérdida bruta

Es la suma de todas las perdidas de cada transaccion (Que haya dado como resultado perdida)

- La ganancia total

Es la diferencia entre la ganancia bruta y la pérdida bruta

- El número de inversiones abiertas

Es el número de inversiones que se compraron pero que no se han vendido aún

- El número de inversiones cerradas

Es el número de transacciones que se compraron y ya se vendieron

- Conclusiones del análisis

Son dos conclusiones, si debe comprar o si se debe vender la acción al día siguiente.

Además se puede visualizar el reporte de inversiones, que consiste en mostrar todas las transacciones que se hubiesen efectuado con sus respectivas fechas de compra y venta y con su ganancia o pérdida.

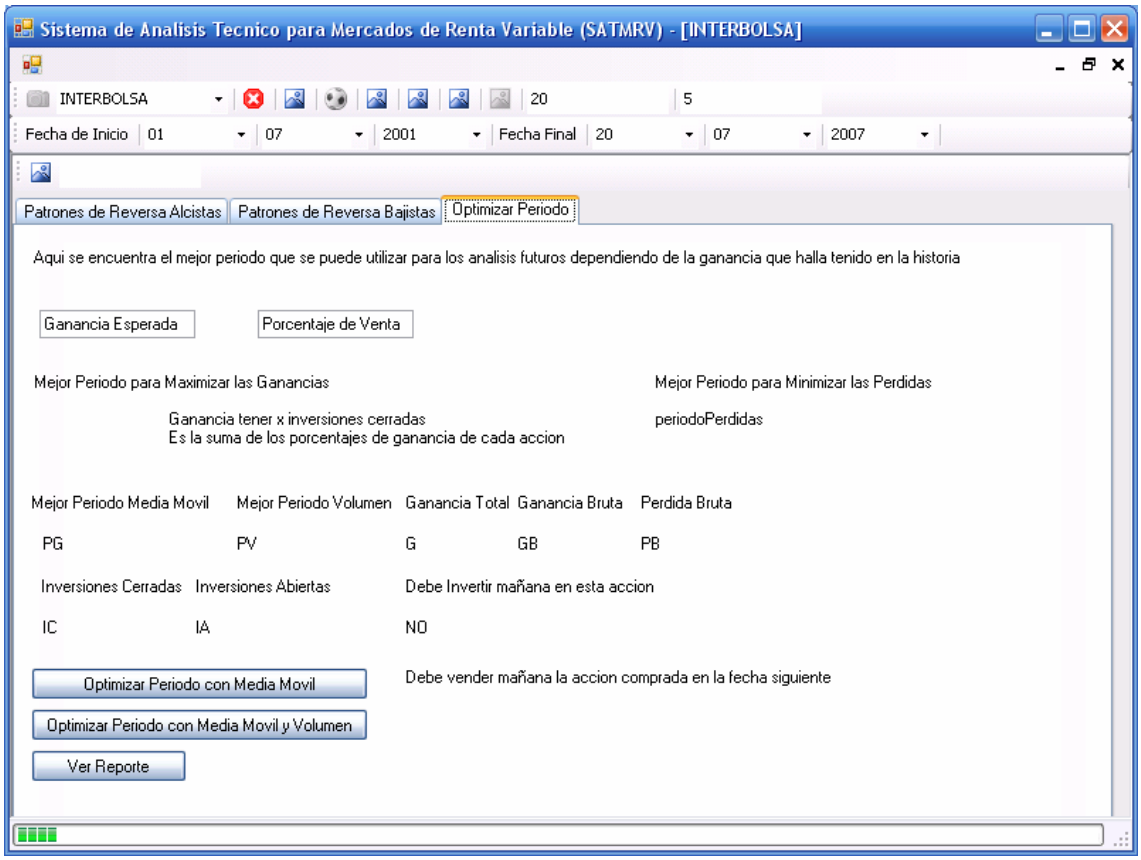

FIGURA 6

El procedimiento para llegar a la conclusión de que se debe invertir en la acción o venderla al día siguiente consiste en llegar al porcentaje de ganancia esperada (que es 20%) por cada acción y un porcentaje de venta por debajo de la ganancia esperada (que es 5%). Cuando se produce una señal de compra se abre una inversión (se avisa que se debe invertir al día siguiente), la cual se mantiene abierta hasta que el precio llega a la ganancia esperada, luego de esto, se debe vender cuando el precio rebaje hasta el porcentaje de venta (se avisa que se debe cerrar la inversión), esto se hace para no vender la acción apenas llegue a su ganancia esperada, ya que puede seguir subiendo, y puede haber una corrección del precio en la cual no se debe vender, la cual es representada por el porcentaje de venta.

Si el precio de la acción sigue subiendo por encima del porcentaje de ganancia esperada, y este llega a un múltiplo de la ganancia esperada, se pone un nuevo punto de venta, con el porcentaje de venta por debajo de este que es el múltiplo de la primera ganancia esperada.

El siguiente es el código fuente de las funciones que se utilizan para optimizar el período de una media móvil como filtro para el análisis de velas japonesas:

```
//Función para optimizar el período
public void optimizarPeriodo()
{
//Listas para almacenar los reportes
      listaReportes = new ArrayList ();
      mejorListaReportes = new ArrayList ();
//Convierte los períodos a los porcentajes para trabajar con ellos
      porcentajeVenta = 1.0 - double.Parse(tbPorcentajeDeVenta.Text) /
                        100;
      gananciaEsperada = double.Parse(tbGananciaEsperada.Text) / 100 +
                        1.0;

//Listas que almacenan los precios y los promedios exponenciales de la
//acción
      ArrayList listaPrecios = new ArrayList();
      listaPrecios = listaPreciosGlobal;
      ArrayList listaPromedios = new ArrayList();
//Variables que almacenan el mejor período del análisis y el número de
//operaciones abiertas y cerradas, además de la mejor ganancia y las
//conclusiones del análisis
       int mejorPeriodo = 1;
       float mejorGanancia = 0;
       int inversionesAbiertas = 0;
       int inversionesCerradas = 0;
       double gananciaBruta = 0;
       double perdidaBruta = 0;
       string fechaCompraParaLaVenta = "";
       string debeInvertirHoy = "";
//Por cada período de 1 a cien halla el que tenga la mejor ganancia,
//las inversiones abiertas y cerradas
       for (int i = 1; i < 100; i++)
       {
//Obtiene los promedios exponenciales con el período i
           listaPromedios =
           obtenerMediaMovilExponencial(i.ToString());
//Función para hallar la ganancia
           float bufferGanacia = hallarGanacia(listaPrecios,
           listaPromedios);
//Si la ganancia es mayor que una obtenida anteriormente, esta se toma
//como la mejor y todos los resultados del análisis con el período i
           if (bufferGanacia > mejorGanancia)
           {
               mejorGanancia = bufferGanacia;
               mejorPeriodo = i;
               inversionesAbiertas = numeroInversionesAbiertas;
               inversionesCerradas = numeroInversionesCerradas;
               gananciaBruta = gananciaGlobal;
               perdidaBruta = perdidaGlobal;
               fechaCompraParaLaVenta = fechaDeCompraParaLaVentaHoy;
               debeInvertirHoy = debeInvertir;
               mejorListaReportes = listaReportes;
           }
       }
//Muestra los datos en el formulario
       lbPeriodoGanancias.Text = mejorPeriodo.ToString();
//Multiplica por cien para mostrar la ganacia como porcentaje
```

```
        mejorGanancia *= 100;
//Añande el signo de porcentaje para el valor
        lbGanancias.Text = mejorGanancia.ToString() + "%";
//Muestra el número de inversiones abiertas y cerradas hasta el
//momento
        lbInversionesAbiertas.Text = inversionesAbiertas.ToString();
        lbInversionesCerradas.Text = inversionesCerradas.ToString();
//Muestra la ganacia y la pérdida brutas en porcentaje y las
//conclusiones del análisis
      gananciaBruta *= 100;
      perdidaBruta *= 100;
      lbGanaciaBruta.Text =
      float.Parse(gananciaBruta.ToString()).ToString() + "%";
      lbPerdidaBruta.Text = float.Parse
      (perdidaBruta.ToString()).ToString () + "%";
      lbDebeInvertir.Text = debeInvertirHoy;
      lbFechaCompra.Text = fechaCompraParaLaVenta;
}



//Función que halla la ganancia de un periodo haciendo filtro de velas
//con una media móvil exponencial
private float hallarGanacia(ArrayList listaPrecios, ArrayList
listaPromedios)
{
//Funcion que halla los patrones de velas japonesas
      hallarPatronesDeVelasJaponesas(listaPrecios);
//verifica que las listas tengan por lo menos 2 elementos
      if (listaPrecios.Count > 2 && listaPromedios.Count > 2)
      {
//Lista que almacena los reportes de las inversiones
            listaReportes = new ArrayList ();
//variables que almacenan la información de la transacción histórica
            double ganancia = 0;
            double perdida = 0;
            double valorInicial = 0;
            double valorFinal = 0;
            gananciaGlobal = 0;
            perdidaGlobal = 0;
            DateTime fechaCompra = new DateTime ();
            DateTime fechaVenta = new DateTime ();
//Variable booleana que indica cuando se puede vender
            bool puedeVender = false;
//Variable que almacena el punto donde se puede vender
            double puntoVenta = 0;
//Variables que almacenan el número de inversiones cerradas y abiertas
//que quedan al final del análisis
            numeroInversionesCerradas = 0;
            numeroInversionesAbiertas = 0;
//Variable que almacena la fecha de compra de la acción que se debe
//vender
            fechaDeCompraParaLaVentaHoy = "";
//Variable que almacena la conclusión de si se debe comprar o no
            debeInvertir = "NO";
//halla las fechas de los patrones de velas alcistas filtrados con una
//media exponencial
            ArrayList fechasAlcistas =
            filtrarFechasAlcistas(listaPatronesReversaAlcistas,
            listaPrecios, listaPromedios);
```

```
//halla las fechas de los patrones de velas bajistas filtrados con una
//media exponencial
            ArrayList fechasBajistas =
            filtrarFechasBajistas(listaPatronesReversaBajistas,
            listaPrecios, listaPromedios);
//Halla la diferencia entre la lista de los precios y la de los
//promedios, ya que la de promedios siempre es menor que la de
//precios, para tomar los días correctos en el análisis
            int diferencia = listaPrecios.Count -
            listaPromedios.Count;

//Halla la ganacia de cada transacción en cada fecha y las suma en la
//variable ganancia
for (int i = 0; i < fechasAlcistas.Count; i++)
{
//estructura dia para ser utilizada como buffer
      PantallaPrincipal.dia dia = new PantallaPrincipal.dia();
//Toma la fecha en la cual se identifica el patron
      fechaCompra = DateTime.Parse(fechasAlcistas[i].ToString());
      fechaVenta = DateTime.Parse("01/01/2100");
//Toma la fecha en la cual se compra la acción que seria al día
//siguiente de identificar el patrón
      for (int j = 0; j < listaPromedios.Count - 1; j++)
      {
            dia = (PantallaPrincipal.dia)listaPrecios[j + diferencia];
            if (dia.fecha == fechaCompra)
            {
                  dia = (PantallaPrincipal.dia)listaPrecios[j +
                  diferencia + 1];
                  fechaCompra = dia.fecha;
//sale del ciclo al encontrar la fecha de compra
                  break;
            }
       }
//Toma el valor inicial de la compra, como si fuera el precio de
//cierre del día siguiente a la identificación del patrón de velas
       valorInicial = dia.valorCierre;
//Pone el punto de venta al porcentaje elegido por encima del precio
//de cierre del día de compra
       puntoVenta = valorInicial * gananciaEsperada;
//Hace a la variable falsa para indicar que todavía no se puede vender
       puedeVender = false;
//Halla la ganancia de la inversión haciendo que se venda luego de que
//sobrepase un punto de venta y vuelva a caer por debajo de él en el
//porcentaje elegido
       for (int j = 0; j < listaPromedios.Count; j++)
       {
//Variable para ser utilizada como buffer
            dia = (PantallaPrincipal.dia)listaPrecios[j + diferencia];
//Toma como fecha de partida la fecha de compra
            if (dia.fecha > fechaCompra)
            {
//Si el día que se analiza sobrepasa el punto de venta
//entonces puede vender
                  if (dia.valorCierre >= puntoVenta)
                  {
                        puedeVender = true;
                  }
//Si puede vender y ha caído por debajo del punto de venta
//se debe vender y se almacena la fecha de venta que sería
//la fecha del día siguiente de identificar la venta
```

```
                    if (puedeVender && dia.valorCierre < puntoVenta *
                    porcentajeVenta)
                    {

//Trata de hallar la fecha del día siguiente
//pero si el día de identificación de venta es el
//último en la lista, entonces avisa que se debe vender
//ese día
                          try
                          {
                                dia =
                                (PantallaPrincipal.dia)listaPrecios[j +
                                diferencia + 1];
                                fechaVenta = dia.fecha;
                                break;
                          }
                          catch
                          {
                                dia =
                                (PantallaPrincipal.dia)listaPrecios[j +
                                diferencia];
                                fechaVenta = dia.fecha;
                                break;
                          }
                     }
//Si puede vender y el precio del día analizado sobrepasa
//un múltiplo del punto de venta se toma como nuevo punto
//de venta
                     if (puedeVender && dia.valorCierre >= puntoVenta *
                     gananciaEsperada)
                     {
                          puntoVenta = puntoVenta * gananciaEsperada;
                     }
               }
          }

//Si la fecha de venta es igual a la fecha de compra
//se hace la fecha de venta muy superior a la de compra
//para así indicar que la inversión debe tomarse como
//abierta
          if (fechaVenta == fechaCompra)
          fechaVenta = DateTime.Parse("01/01/2100");
//Si la fecha de venta dá como resultado el día de hoy
//pero todavía no se puede vender, se hace la fecha de venta
//muy superior a la de hoy para así indicar que la inversión
//debe tomarse como abierta
          if (fechaVenta == DateTime.Today && puedeVender == false)
          {
                  fechaVenta = DateTime.Parse("01/01/2100");
          }
//Hace que el valor incial y el final se les quite un 0.5% de su valor
//que sería la comision que se debe pagar al comisionista por la
//transacción
            valorFinal = dia.valorCierre;
            valorInicial = valorInicial * 1.005;
            valorFinal = valorFinal * 0.995;
//toma el total de las compras para luego restarle las
//inversiones abiertas al final y encontrar así las
//inversiones cerradas
            numeroInversionesCerradas++;
//Aquí decide si la ganacia se toma en el caso de que se halla cerrado
```

```
//la transacción en la fecha del día de hoy
            dia =
            (PantallaPrincipal.dia)listaPrecios[listaPrecios.Count -
            1];
            if (dia.fecha > fechaVenta)
            {
                  if (valorInicial < valorFinal)
                  {
                        ganancia += (valorFinal / valorInicial) - 1;
                  }
                  else
                  {
                        perdida += 1 - (valorFinal / valorInicial);
                  }
                  double gananciaIndividual = (valorFinal /
                  valorInicial) - 1;
                  anadirAlReporte(this.Text, "cerrada",
                  fechaCompra.ToString(), fechaVenta.ToString(),
                  gananciaIndividual.ToString());

            }
//Si no se ha cerrado la transacción entonces se almacena en la
//variable que cuenta las transacciones no cerradas, y se decrementa
//la variable que tiene las acciones cerradas
            else
            {
                  PantallaPrincipal.dia buffer =
                  (PantallaPrincipal.dia)listaPrecios[listaPrecios.Coun
                  t - 1];
                  float gananciaIndividual = (buffer.valorCierre /
                  float.Parse(valorInicial.ToString())) - 1;
                  anadirAlReporte (this.Text
                  ,"abierta",fechaCompra.ToString (),"no vendida",
                  gananciaIndividual.ToString ());
                  numeroInversionesAbiertas++;
                  if(numeroInversionesCerradas >0)
                        numeroInversionesCerradas--;
            }
      }
//Retorna el valor de la ganacia como un float
      gananciaGlobal = ganancia;
      perdidaGlobal = perdida;
      ganancia = ganancia - perdida;

//Si la fecha de venta es la de hoy se almacena la fecha
//de compra de la acción para indicar cual es la que se
//debe vender hoy
      if (fechaVenta >= DateTime.Today && fechaVenta != DateTime.Parse
      ("01/01/2100") && fechaVenta > fechaCompra)
      {
            fechaDeCompraParaLaVentaHoy = fechaCompra.ToString();
      }
//Si la fecha de compra es hoy se hace a la variable
//positiva para indicar que hoy se debe comprar la acción
      if (fechaCompra >= DateTime.Today)
      {
            debeInvertir = "SI";
      }
//retorna la ganancia
      return float.Parse(ganancia.ToString());
      }
```

```
      return 0;
}


//Función para filtrar los patrones de velas japonesas alcistas
//con la media exponencial indicada
private ArrayList filtrarFechasAlcistas(ArrayList
listaPatronesReversaAlcistasLocal, ArrayList listaPrecios, ArrayList
listaPromedios)
{
      int diferencia = listaPrecios.Count - listaPromedios.Count;
//Filtra las fechas en los patrones de reversa alcistas que no estén
//en una tendencia a la baja
      for (int i = 0; i < listaPromedios.Count; i++)
      {
            PantallaPrincipal.dia dia =
            (PantallaPrincipal.dia)listaPrecios[i + diferencia];
            float promedio = (float)listaPromedios[i];
            if (dia.valorCierre > promedio * 1.2)
            {
                  if
                  (listaPatronesReversaAlcistasLocal.Contains(dia.fecha
                  ))
                        listaPatronesReversaAlcistasLocal.Remove(dia.fe
                        cha);
            }
      }
//Retorna la lista de fechas de patrones alcistas filtrados
      return listaPatronesReversaAlcistasLocal;
}


//Funcion para filtrar los patrones de velas japonesas bajistas
//con la media exponencial indicada
private ArrayList filtrarFechasBajistas(ArrayList
listaPatronesReversaBajistasLocal, ArrayList listaPrecios, ArrayList
listaPromedios)
{
      int diferencia = listaPrecios.Count - listaPromedios.Count;
//Fltra las fechas en los patrones de reversa bajistas que no estén en
//una tendencia a la alza
      for (int i = 0; i < listaPromedios.Count; i++)
      {
            PantallaPrincipal.dia dia =
            (PantallaPrincipal.dia)listaPrecios[i + diferencia];
            float promedio = (float)listaPromedios[i];
            if (dia.valorCierre < promedio * 0.8)
            {
                  if
                  (listaPatronesReversaBajistasLocal.Contains(dia.fecha
                  ))

                        listaPatronesReversaBajistasLocal.Remove(dia.fe
                        cha);
            }
      }
//Retorna las fechas de patrones bajistas filtrados
      return listaPatronesReversaBajistasLocal;
}
```

```
//Función para obtener los promedios móviles exponenciales
private ArrayList obtenerMediaMovilExponencial(string periodo)
{
//Arraylist para almacenar los promedios móviles de la acción
      ArrayList listaPromedios = new ArrayList();
//Convierte el período a entero
      int periodoNumerico = int.Parse(periodo);
//verifica que el período sea mayor que cero
      if (periodoNumerico > 0)
      {
//Arraylist para almacenar los precios diarios de la acción
            ArrayList listaPrecios = new ArrayList();
            listaPrecios = listaPreciosGlobal;
//Variables buffer para almacenar los promedios de un día y el
//anterior
            float promedio = 0;
            float promedioAnterior = 0;
//Constante para hacer las operaciones
            float dos = 2;
//Obtenemos los promedios moviles
            for (int i = periodoNumerico - 1; i < listaPrecios.Count;
                  i++)
            {
//Variable buffer para almacenar la información de un día
                  PantallaPrincipal.dia buffer =
                  (PantallaPrincipal.dia)listaPrecios[i];
//Si es el primer valor del cual se va a hallar el promedio
//exponencial se llama a la función
//primerPromedioMovilExponencialNoRecursivo para hacerlo
                  if (i == periodoNumerico - 1)
                  {
//funcion que halla el primer promedio exponencial
                        promedio =
                        primerPromedioMovilExponencialNoRecursivo(lista
                        Precios, periodoNumerico);
                  }
//Si no es el primer valor entonces se halla con la fórmula del día en
//función del día anterior
                  else
                  {
                        promedio = promedioAnterior + (dos /
                        (periodoNumerico + 1)) * (buffer.valorCierre -
                        promedioAnterior);
                  }
//Añade el promedio a la lista de promedios exponenciales
                  listaPromedios.Add(promedio);
//Guarda el valor del día actual para utilizarlo en el día siguiente
                  promedioAnterior = promedio;
            }
      }
//Devuelve la lista de promedios exponenciales
      return listaPromedios;
}
```

```
//Funcion para obtener el primer promedio móvil exponencial
private float primerPromedioMovilExponencialNoRecursivo(ArrayList
listaPrecios, int periodo)
{
//Se utiliza la fórmula:
//ME(t) = ME(t-1) + F * (P - ME(t-1))
//F=2 / (N+1)
//Se asigna el primer día de la lista de los precios en esta variable
      PantallaPrincipal.dia buffer =
      (PantallaPrincipal.dia)listaPrecios[0];
//Variable que almacena el resultado de un promedio móvil exponencial
      float MEX = 0;
//Variable que almacena el resultado de el promedio móvil exponencial
//inmediatamente anterior a MEX al principio se le pone el valor de
//cierre del primer día ya que éste no tiene promedio exponencial
      float MEXMenosUno = buffer.valorCierre;
//Constante para hacer los cálculos
      float dos = 2;
//Se halla el primer promedio móvil exponencial aplicando la fórmula
//tantas veces como indique el período
      for (int i = 1; i < periodo; i++)
      {
            buffer = (PantallaPrincipal.dia)listaPrecios[i];
            MEX = MEXMenosUno + dos / (periodo + 1) *
            (buffer.valorCierre - MEXMenosUno);
            MEXMenosUno = MEX;
      }
//retorna el valor del primer promedio móvil exponencial
      return MEX;
}
```

CAPITULO 5: OPTIMIZACION DE PERIODO DE MEDIA MOVIL EXPONENCIAL Y DE VOLUMEN COMO FILTROS DE PATRONES DE REVERSA DE VELAS JAPONESAS

"El volumen es el número de entidades que han sido objeto de operaciones durante el período de estudio" (Murphy 2000).

Cuando se realiza una transacción en la Bolsa de Valores, el volumen de esta representa la fuerza con la que se puede consolidar la tendencia. Ya que las señales que se producen por medio de los patrones de reversa de velas japonesas filtradas con media móvil pueden tener volúmenes altos o bajos, se hace necesario tener un filtro adicional al de la media móvil exponencial para que las señales resultantes sean fuertes, o sea, con un volumen alto.

Este filtro consiste en aplicar una media móvil exponencial para el volumen, y si el valor del volumen del día es mayor por encima del 1% del promedio móvil exponencial, dicho volumen se considera alto.

El período óptimo de la media móvil exponencial aplicada al volumen se halla haciendo iteraciones del período de 1 a 100 por cada período de la media móvil que se aplica a los precios de cierre, optimizando no solo el periodo de la media del volumen, sino también el periodo de la media móvil de los precios de cierre hallando la combinación que haya dado como resultado mas ganancia en el período histórico analizado.

En la figura 6 se puede visualizar los resultados que arroja este análisis, que al igual que el filtro descrito el en capítulo anterior tiene los mismos resultados, que son el periodo óptimo de media móvil de precios de cierre, la ganancia bruta, la pérdida bruta, la ganancia total, el número de inversiones abiertas y cerradas y si debe comprar o vender la acción al día siguiente, con el resultado adicional del período de la media móvil exponencial para el volumen.

El siguiente es el código fuente de las funciones utilizadas para realizar este análisis:

```
//Función para optimizar los patrones de velas japonesas con volumen
//y con media móvil exponencial
public void optimizarPeriodoConVolumenYMediaExponencial()
{
//Listas que almacenan los reportes
      listaReportes = new ArrayList();
      mejorListaReportes = new ArrayList();
//Convierte los períodos a los porcentajes para trabajar con ellos
      porcentajeVenta = 1.0 - double.Parse(tbPorcentajeDeVenta.Text) /
                        100;
      gananciaEsperada = double.Parse(tbGananciaEsperada.Text) / 100 +
                        1.0;
//Listas que almacenan los precios y los preomedios exponenciales de
//la acción
      ArrayList listaPrecios = new ArrayList();
      listaPrecios = listaPreciosGlobal;
      ArrayList listaPromedios = new ArrayList();
      ArrayList listaPromediosVolumen = new ArrayList();
//Variables que almacenan el mejor período del análisis y el número de
//operaciones abiertas y cerradas, además de la mejor ganancia y las
//conclusiones del análisis
      int mejorPeriodoMediaMovil = 1;
      int mejorPeriodoVolumen = 1;
      float mejorGanancia = 0;
      int inversionesAbiertas = 0;
      int inversionesCerradas = 0;
      double gananciaBruta = 0;
      double perdidaBruta = 0;
      string fechaCompraParaLaVenta = "";
      string debeInvertirHoy = "";
//Por cada período de 1 a cien halla el que tenga la mejor ganancia y
//las inversiones abiertas y cerradas
//Crea un objeto de la clase GraficaVolumen para hallar los volúmenes
      GraficaVolumen filtrarConVolumen = new GraficaVolumen();
      for (int i = 1; i < 100; i++)
      {
//Obtiene la media móvil exponencial de los precios de cierre
          listaPromedios = obtenerMediaMovilExponencial(i.ToString());
          for (int j = 1; j < 100; j++)
          {
//Si la ganancia es mayor que una obtenida anteriormente se toma
//como la mejor y todos los resultados del análisis con el periodo i
//para la media móvil de los precios de cierre y período j para
//la media móvil del volumen
                filtrarConVolumen.listaPreciosGlobal =
                listaPreciosGlobal;
                filtrarConVolumen.Text = this.Text;
//Obtiene la media móvil exponencial del volumen
                listaPromediosVolumen =
                filtrarConVolumen.obtenerMediaMovilExponencial(j.ToStr
                ing());
//Halla la ganancia
                float bufferGanacia =
                hallarGananciaConVolumen(listaPrecios, listaPromedios,
                listaPromediosVolumen);
                if (bufferGanacia > mejorGanancia)
                {
                    mejorGanancia = bufferGanacia;
                    mejorPeriodoMediaMovil = i;
                    mejorPeriodoVolumen = j;
                    inversionesAbiertas = numeroInversionesAbiertas;
```

```
                    inversionesCerradas = numeroInversionesCerradas;
                    gananciaBruta = gananciaGlobal;
                    perdidaBruta = perdidaGlobal;
                    fechaCompraParaLaVenta =
                    fechaDeCompraParaLaVentaHoy;
                    debeInvertirHoy = debeInvertir;
                    mejorListaReportes = listaReportes;
                }

            }
        }
//Muestra los datos en el formulario
        lbPeriodoGanancias.Text = mejorPeriodoMediaMovil.ToString();
        lbPeriodoVolumen.Text = mejorPeriodoVolumen.ToString();
        //multiplica por cien para mostrar la ganacia como porcentaje
        mejorGanancia *= 100;
        //Añande el signo de porcentaje para el valor
        lbGanancias.Text = mejorGanancia.ToString() + "%";
        //Muestra el número de inversiones abiertas y cerradas hasta
        el momento
        lbInversionesAbiertas.Text = inversionesAbiertas.ToString();
        lbInversionesCerradas.Text = inversionesCerradas.ToString();
        //Mueatra la ganacia y la pérdida bruta en porcentaje y las
        //conclusiones del análisis
        gananciaBruta *= 100;
        perdidaBruta *= 100;
        lbGanaciaBruta.Text =
        float.Parse(gananciaBruta.ToString()).ToString() + "%";
        lbPerdidaBruta.Text =
        float.Parse(perdidaBruta.ToString()).ToString() + "%";
        lbDebeInvertir.Text = debeInvertirHoy;
        lbFechaCompra.Text = fechaCompraParaLaVenta;
}



//Función para hallar la ganancia con los 2 filtros
private float hallarGananciaConVolumen(ArrayList listaPrecios,
ArrayList listaPromedios, ArrayList listaPromediosVolumen)
    {
//Función que halla los patrones de velas japonesas
        hallarPatronesDeVelasJaponesas(listaPrecios);
//verifica que las listas no tengan por lo menos 2 elementos
        if (listaPrecios.Count > 2 && listaPromedios.Count > 2)
        {
            listaReportes = new ArrayList();
//variables que almacenan la información de la transacción histórica
            double ganancia = 0;
            double perdida = 0;
            double valorInicial = 0;
            double valorFinal = 0;
            gananciaGlobal = 0;
            perdidaGlobal = 0;
            DateTime fechaCompra = new DateTime();
            DateTime fechaVenta = new DateTime();
//Variable booleana que indica cuando se puede vender
            bool puedeVender = false;
//Variable que almacena el punto donde se puede vender
            double puntoVenta = 0;
//Variables que almacenan el número de inversiones cerradas y abiertas
//que quedan al final del análisis
```

```
            numeroInversionesCerradas = 0;
            numeroInversionesAbiertas = 0;
//Variable que almacena la fecha de compra de la acción que se debe
//vender
            fechaDeCompraParaLaVentaHoy = "";
//Variable que almacena la conclusión de si se debe comprar o no
            debeInvertir = "NO";
//halla las fechas de los patrones de velas alcistas filtrados con una
//media exponencial
            ArrayList fechasAlcistas =
            filtrarFechasAlcistas(listaPatronesReversaAlcistas,
            listaPrecios, listaPromedios);
//halla las fechas de los patrones de velas bajistas filtrados con una
//media exponencial
            ArrayList fechasBajistas =
            filtrarFechasBajistas(listaPatronesReversaBajistas,
            listaPrecios, listaPromedios);

//Filtra los patrones con volumen
            fechasAlcistas =
            filtrarFechasAlcistasConVolumen(listaPatronesReversaAlcist
            as, listaPrecios, listaPromediosVolumen);
            fechasBajistas =
            filtrarFechasBajistasConVolumen(listaPatronesReversaBajist
            as, listaPrecios, listaPromediosVolumen);
//Halla la ganacia de cada transacción en cada fecha y las suma en la
//variable ganancia
            int diferencia = listaPrecios.Count -
            listaPromedios.Count;
            for (int i = 0; i < fechasAlcistas.Count; i++)
            {
//estructura dia para ser utilizada como buffer
                PantallaPrincipal.dia dia = new
                PantallaPrincipal.dia();
                //Toma la fecha en la cual se identifica el patrón
                fechaCompra =
                DateTime.Parse(fechasAlcistas[i].ToString());
                fechaVenta = DateTime.Parse("01/01/2100");
//Toma la fecha en la cual se compra la acción que sería al día
//siguiente de identificar el patrón
                for (int j = 0; j < listaPromedios.Count - 1; j++)
                {
                    dia = (PantallaPrincipal.dia)listaPrecios[j +
                    diferencia];
                    if (dia.fecha == fechaCompra)
                    {
                        dia = (PantallaPrincipal.dia)listaPrecios[j +
                        diferencia + 1];
                        fechaCompra = dia.fecha;
//sale del ciclo al encontrar la fecha de compra
                        break;
                    }
                }

//Toma el valor inicial de la compra, como si fuera el precio de
//cierre del día siguiente a la identificación del patrón de velas
                valorInicial = dia.valorCierre;
//Pone el punto de venta al porcentaje elegido por encima del precio
//de cierre del día de compra
                puntoVenta = valorInicial * gananciaEsperada;
//Hace a la variable falsa para indicar que todavía no se puede vender
```

```
                puedeVender = false;
//Halla la ganancia de la inversión haciendo que se venda luego de que
//sobrepase un punto de venta y vuelva a caer por debajo de él en el
//porcentaje elegido
                for (int j = 0; j < listaPromedios.Count; j++)
                {
//Variable para ser utilizada como buffer
                  dia = (PantallaPrincipal.dia)listaPrecios[j +
                  diferencia];
//Toma como fecha de partida la fecha de compra
                  if (dia.fecha > fechaCompra)
                  {
//Si el día que se analiza sobrepasa el punto de venta
//entonces puede vender
                      if (dia.valorCierre >= puntoVenta)
                      {
                          puedeVender = true;
                      }
//Si puede vender y ha caído por debajo del punto de venta
//se debe vender y se almacena la fecha de venta que sería
//la fecha del día siguiente de identificar la venta
                      if (puedeVender && dia.valorCierre < puntoVenta
                      * porcentajeVenta)
                      {
//Trata de hallar la fecha del día siguiente
//pero sí el día de identificación de venta es el
//último en la lista entonces avisa que se debe vender
//ese día
                          try
                          {
                                dia =
                                (PantallaPrincipal.dia)listaPrecios[
                                j + diferencia + 1];
                                fechaVenta = dia.fecha;
                                break;
                          }
                          catch
                          {
                               dia =
                               (PantallaPrincipal.dia)listaPrecios[j
                                + diferencia];
                               fechaVenta = dia.fecha;
                               break;
                          }
                       }
//Si puede vender y el precio del día analizado sobrepasa
//un múltiplo del punto de venta se toma como nuevo punto
//de venta
                      if (puedeVender && dia.valorCierre >=
                      puntoVenta * gananciaEsperada)
                      {
                             puntoVenta = puntoVenta *
                             gananciaEsperada;
                      }
                  }
            }

//Si la fecha de venta es igual a la fecha de compra
//se hace la fecha de venta muy superior a la de compra
//para así indicar que la inversión debe tomarse como
//abierta
```

```
            if (fechaVenta == fechaCompra)
                fechaVenta = DateTime.Parse("01/01/2100");
//Si la fecha de venta dá como resultado el día de hoy
//pero todavía no se puede vender, se hace la fecha de venta
//muy superior a la de hoy para así indicar que la inversión
//debe tomarse como abierta
            if (fechaVenta == DateTime.Today && puedeVender == false)
            {
                fechaVenta = DateTime.Parse("01/01/2100");
            }
//Hace que el valor incial y el final se les quite un 0.5% de su valor
//que sería la comisión que se debe pagar al comisionista por la
//transacción
                valorFinal = dia.valorCierre;
                valorInicial = valorInicial * 1.005;
                valorFinal = valorFinal * 0.995;


//toma el total de las compras para luego restarle las
//inversiones abiertas al final y encontrar así las
//inversiones cerradas
                numeroInversionesCerradas++;
//Aquí decide si la ganacia se toma en el caso de que se halla cerrado
//la transaccion en la fecha del día de hoy
                dia =
                (PantallaPrincipal.dia)listaPrecios[listaPrecios.Count
                 - 1];
                if (dia.fecha > fechaVenta)
                {
                    if (valorInicial < valorFinal)
                    {
                        ganancia += (valorFinal / valorInicial) - 1;
                    }
                    else
                    {
                        perdida += 1 - (valorFinal / valorInicial);
                    }
//Añade la ganacia individual al reporte
                    double gananciaIndividual = (valorFinal /
                    valorInicial) - 1;
                    anadirAlReporte(this.Text, "cerrada",
                    fechaCompra.ToString(), fechaVenta.ToString(),
                    gananciaIndividual.ToString());
                }
//Si no se ha cerrado la transacción entonces se almacena en la
//variable que cuenta las transacciones abiertas, y se decrementa
//la variable que tiene las acciones cerradas
                else
                {
                  PantallaPrincipal.dia buffer =
                  (PantallaPrincipal.dia)listaPrecios[listaPrecios.Cou
                  nt - 1];
                  float gananciaIndividual = (buffer.valorCierre /
                  float.Parse(valorInicial.ToString())) - 1;
                  anadirAlReporte(this.Text, "abierta",
                  fechaCompra.ToString(), "no vendida",
                  gananciaIndividual.ToString());
                  numeroInversionesAbiertas++;
                  if (numeroInversionesCerradas > 0)
                      numeroInversionesCerradas--;
                }
```

```
        }
//Retorna el valor de la ganacia como un float
        gananciaGlobal = ganancia;
        perdidaGlobal = perdida;
        ganancia = ganancia - perdida;
//Si la fecha de venta es la de hoy se almacena la fecha
//de compra de la acción para indicar cual es la que se
//debe vender hoy
        if (fechaVenta >= DateTime.Today && fechaVenta !=
        DateTime.Parse("01/01/2100") && fechaVenta > fechaCompra)
        {
            fechaDeCompraParaLaVentaHoy = fechaCompra.ToString();
        }
//Si la fecha de compra es hoy se hace a la variable
//positiva para indicar que hoy se debe comprar la acción
        if (fechaCompra >= DateTime.Today)
        {
              debeInvertir = "SI";
        }
        //retorna la ganancia
        return float.Parse(ganancia.ToString());
    }
    return 0;
}
```

```
//Función para filtrar las fechas alcistas con volumen
private ArrayList filtrarFechasAlcistasConVolumen(ArrayList
listaPatronesReversaAlcistasLocal, ArrayList listaPrecios, ArrayList
listaPromedios)
{
//Filtra las fechas en los patrones de reversa alcistas que no tengan
//el volumen suficiente
//Encuentra cual lista tiene menor tamaño para luego empezar a leer
//desde el elemento correcto
        int minimoTamaño = 0;
        if (listaPrecios.Count < listaPromedios.Count)
        {
            minimoTamaño = listaPrecios.Count;
        }
        else
        {
            minimoTamaño = listaPromedios.Count;
        }

//halla ladiferencia entre los tamaños de las listas
        for (int i = 0; i < minimoTamaño; i++)
        {
            int diferencia = 0;
//Si la lista de promedios es menor que la de precios
//se halla la diferencia y se toma el día de lista precios
//que concuerda con el primero de la lista de promedios
            if (listaPromedios.Count < listaPrecios.Count)
            {
                diferencia = listaPrecios.Count - listaPromedios.Count;
                PantallaPrincipal.dia dia =
                (PantallaPrincipal.dia)listaPrecios[i + diferencia];
//Se filtra la fecha si no cumple con el volumen necesario
//que es el uno por ciento mayor que el de la media
```

```
                float promedio = (float)listaPromedios[i];
                if (dia.volumen < promedio * 1.01)
                {
                    if
                    (listaPatronesReversaAlcistasLocal.Contains(dia.fech
                    a))
                    listaPatronesReversaAlcistasLocal.Remove(dia.fecha);
                }
            }
//Si la lista de promedios es mayor que la de precios
//se halla la diferencia y se toma el día de lista promedios
//que concuerda con el primero de la lista de precios
          else
          {
               diferencia = listaPromedios.Count - listaPrecios.Count;
               PantallaPrincipal.dia dia =
               (PantallaPrincipal.dia)listaPrecios[i];

//Se filtra la fecha si no cumple con el volumen necesario
//que es el uno por ciento mayor que el de la media
                float promedio = (float)listaPromedios[i + diferencia];
                if (dia.volumen < promedio * 1.01)
                {
                   if
                   (listaPatronesReversaAlcistasLocal.Contains(dia.fech
                   a))
                   listaPatronesReversaAlcistasLocal.Remove(dia.fecha);
                }
            }
      }
//Retorna la lista de fechas de patrones alcistas filtrados
     return listaPatronesReversaAlcistasLocal;
}



//Función para filtrar las fechas bajistas con volumen
private ArrayList filtrarFechasBajistasConVolumen(ArrayList
listaPatronesReversaBajistasLocal, ArrayList listaPrecios, ArrayList
listaPromedios)
{
//Filtra las fechas en los patrones de reversa bajistas que no tengan
//el volumen suficiente
//Encuentra cual lista tiene menor tamaño para luego empezar a leer
//desde el elemento correcto
      int minimoTamaño = 0;
      if (listaPrecios.Count < listaPromedios.Count)
      {
          minimoTamaño = listaPrecios.Count;
      }
      else
      {
          minimoTamaño = listaPromedios.Count;
      }
//halla la diferencia entre los tamaños de las listas
      for (int i = 0; i < minimoTamaño; i++)
      {
          int diferencia = 0;
//Si la lista de promedios es menor que la de precios
//se halla la diferencia y se toma el día de lista precios
//que concuerda con el primero de la lista de promedios
```

```
            if (listaPromedios.Count < listaPrecios.Count)
            {
                diferencia = listaPrecios.Count - listaPromedios.Count;
                PantallaPrincipal.dia dia =
                (PantallaPrincipal.dia)listaPrecios[i + diferencia];
//Se filtra la fecha si no cumple con el volumen necesario
//que es el uno por ciento mayor que el de la media
                float promedio = (float)listaPromedios[i];
                if (dia.volumen < promedio * 1.01)
                {
                    if
                    (listaPatronesReversaBajistasLocal.Contains(dia.fech
                    a))
                    listaPatronesReversaBajistasLocal.Remove(dia.fecha);
                }
            }


//Si la lista de promedios es mayor que la de precios
//se halla la diferencia y se toma el día de lista promedios
//que concuerda con el primero de la lista de precios
            else
            {
                diferencia = listaPromedios.Count - listaPrecios.Count;
                PantallaPrincipal.dia dia =
               (PantallaPrincipal.dia)listaPrecios[i];
//Se filtra la fecha si no cumple con el volumen necesario
//que es el uno por ciento mayor que el de la media
                float promedio = (float)listaPromedios[i + diferencia];
                if (dia.volumen < promedio * 1.01)
                {
                  if
                  (listaPatronesReversaBajistasLocal.Contains(dia.fecha))
                   listaPatronesReversaBajistasLocal.Remove(dia.fecha);
                }
            }
    }
//Retorna las fechas de patrones bajistas filtrados
    return listaPatronesReversaBajistasLocal;
}
```

CAPITULO 6: OPTIMIZACION DE PERIODOS DE 2 MEDIAS MOVILES EXPONENCIALES PARA HALLAZGO DE PATRONES DE COMPRA Y VENTA

"La media móvil es esencialmente una forma de seguir una tendencia. Su propósito es hacer saber o indicar que ha comenzado una tendencia nueva o que una vieja ha finalizado o ha cambiado de dirección" (Murphy 2000).

Una técnica muy utilizada es la técnica del doble cruce, con la cual se pueden generar señales de compra y de venta a partir del cruce de 2 medias móviles. Cuando la media móvil de corto plazo cruza por encima del 1% de la media de largo plazo, esto se identifica como un cambio de tendencia al alza en los precios, lo cual indica que se debe comprar, mientras que un cruce de la media móvil de corto plazo por debajo del 1% de la media móvil de largo plazo indica lo contrario, vender.

Para identificar los patrones de compra y los de venta a partir del método del doble cruce, se toman las combinaciones de períodos de largo y corto plazo, para hallar el que más utilidad haya tenido en la historia de la acción, para seguir aplicando esa combinación en el futuro cercano.

En la figura 7 se pueden ver los resultados de este análisis que son el período para la media móvil de largo plazo, el período para la media móvil de corto plazo, el número de inversiones abiertas, el número de inversiones cerradas, la ganancia total, la pérdida total y la ganancia neta.

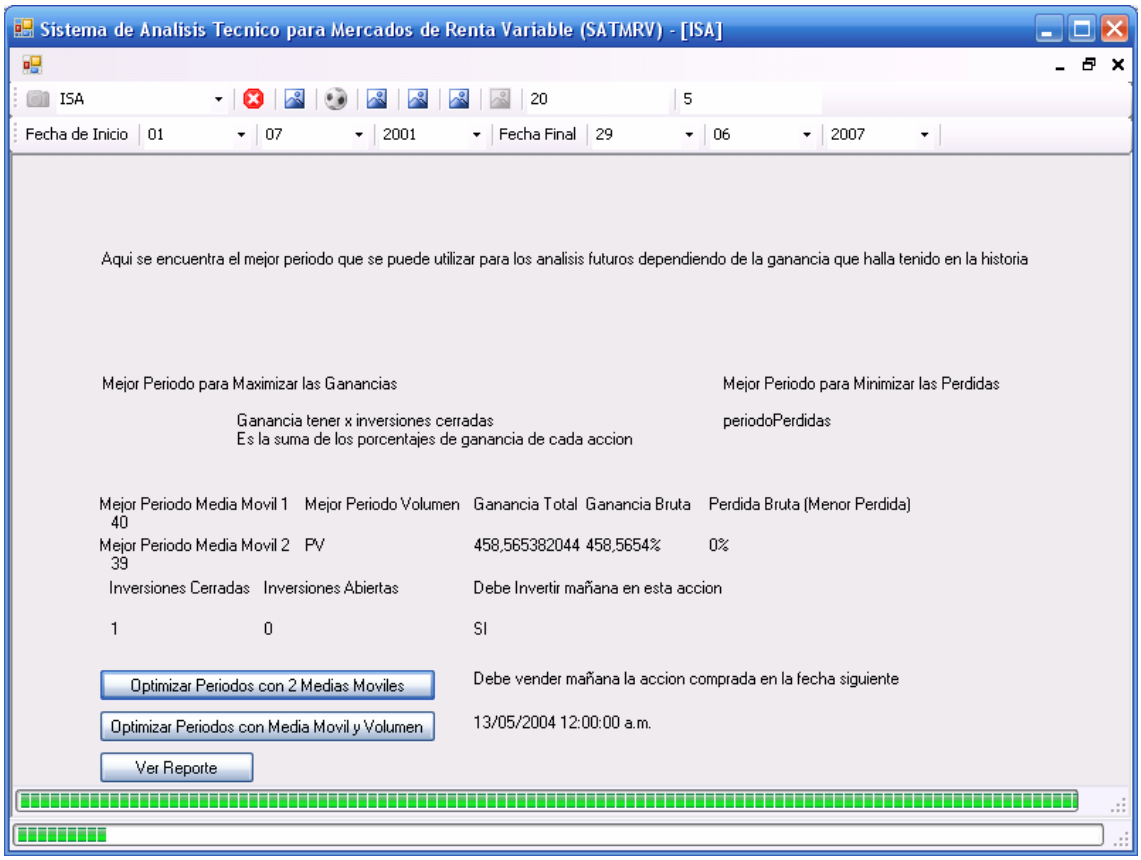

FIGURA 7.

El siguiente es el código fuente de las funciones utilizadas para hallar los patrones de compra y venta por el método del doble cruce:

```
//Función que optimiza los períodos con la mínima pérdida
public void optimizarPeriodo()
{

      listaReportes = new ArrayList();
      mejorListaReportes = new ArrayList();
//Listas que almacenan los precios y los 2 promedios exponenciales de
//la acción y los precios diarios de la acción
      ArrayList listaPrecios = new ArrayList();
      listaPrecios = listaPreciosGlobal;
      ArrayList listaPromedios1 = new ArrayList();
      ArrayList listaPromedios2 = new ArrayList();
//Variables que almacenan el mejor período del análisis y el número de
//operaciones abiertas y cerradas, además de la mejor ganancia y las
//conclusiones del análisis
      int mejorPeriodo1 = 1;
      int mejorPeriodo2 = 1;
      float menorPerdida = 1000000;
      double mayorGanancia = 0;
      int inversionesAbiertas = 0;
      int inversionesCerradas = 0;
      double gananciaBruta = 0;
      double perdidaBruta = 0;
      string fechaCompraParaLaVenta = "";
      string debeInvertirHoy = "";
```

```
      pbProgreso.Maximum = 100;
      pbProgreso.Minimum = 0;
//Por cada período de 1 a cien halla el de largo plazo que tenga la
//mejor ganacia, las inversiones abiertas y cerradas
      for (int i = 1; i < 100; i++)
      {
            listaPromedios1 =
            obtenerMediaMovilExponencial(i.ToString());
//Por cada promedio de corto plazo halla el que tenga mejor ganancia
            for (int j = 1; j < i; j++)
            {
                  listaPromedios2 =
                  obtenerMediaMovilExponencial(j.ToString());
                  float bufferGanancia = hallarGanancia(listaPrecios,
                  listaPromedios1, listaPromedios2);
                  if (bufferGanancia > mayorGanancia)
                  {
                        mayorGanancia = bufferGanancia;
                        mejorPeriodo1 = i;
                        mejorPeriodo2 = j;
                        inversionesAbiertas =
                        numeroInversionesAbiertas;
                        inversionesCerradas =
                        numeroInversionesCerradas;
                        gananciaBruta = gananciaGlobal;
                        perdidaBruta = perdidaGlobal;
                        fechaCompraParaLaVenta =
                        fechaDeCompraParaLaVentaHoy;
                        debeInvertirHoy = debeInvertir;
                        mejorListaReportes = listaReportes;
                  }

             }
             pbProgreso.Value = i;
        }
//Muestra los datos en el formulario
        lbPeriodoGanancias1.Text = mejorPeriodo1.ToString();
        lbPeriodoGanancias2.Text = mejorPeriodo2.ToString();

//multiplica por cien para mostrar la ganacia como porcentaje
        menorPerdida *= 100;
//Añande el signo de porcentaje para el valor
        gananciaBruta *= 100;
        perdidaBruta *= 100;
        double ganancia = gananciaBruta - perdidaBruta;
        lbGanancias.Text = ganancia.ToString() + "%";
//Muestra el número de inversiones abiertas y cerradas hasta el
//momento
        lbInversionesAbiertas.Text = inversionesAbiertas.ToString();
        lbInversionesCerradas.Text = inversionesCerradas.ToString();
//Muestra la ganacia y la pérdida brutas en porcentaje y las
//conclusiones del análisis
        lbGanaciaBruta.Text =
        float.Parse(gananciaBruta.ToString()).ToString() + "%";
        lbPerdidaBruta.Text =
        float.Parse(perdidaBruta.ToString()).ToString() + "%";
        lbDebeInvertir.Text = debeInvertirHoy;
        lbFechaCompra.Text = fechaCompraParaLaVenta;
        try
        {
            PantallaPrincipal.dia hoy =
```

```
            (PantallaPrincipal.dia)listaPrecios[listaPrecios.Count -
            1];
            fechaDeHoy = hoy.fecha;
            precioDeHoy = hoy.valorCierre;
            PantallaPrincipal.dia ayer =
            (PantallaPrincipal.dia)listaPrecios[listaPrecios.Count -
            2];
            precioDeAyer = ayer.valorCierre;
      }
      catch { }
}


//Función para hallar la ganancia
private float hallarGanancia(ArrayList listaPrecios, ArrayList
listaPromediosLargoPlazo, ArrayList listaPromediosCortoPlazo)
{
//Función que halla los patrones de medias móviles
      hallarPatronesDeMediasMoviles(listaPrecios,
      listaPromediosLargoPlazo, listaPromediosCortoPlazo);
//verifica que las listas no sean nulas
      if (listaPrecios.Count > 2 && listaPromediosLargoPlazo.Count > 2
      && listaPromediosCortoPlazo .Count > 2)
      {
            listaReportes = new ArrayList();
//variables que almacenan la información de la transacción histórica
            double ganancia = 0;
            double perdida = 0;
            double valorInicial = 0;
            double valorFinal = 0;
            gananciaGlobal = 0;
            perdidaGlobal = 0;
            DateTime fechaCompra = new DateTime();
            DateTime fechaVenta = new DateTime();
//Variables que almacenan el número de inversiones cerradas y abiertas
//que quedan al final del análisis
            numeroInversionesCerradas = 0;
            numeroInversionesAbiertas = 0;
//Variable que almacena la fecha de compra de la acción que se debe
//vender
            fechaDeCompraParaLaVentaHoy = "";
//Variable que almacena la conclusión de si se debe comprar o no
            debeInvertir = "NO";
//Se inicializan los promedios y el día con el primer elemento de la
//lista
            int diferencia1 = listaPrecios.Count -
            listaPromediosLargoPlazo.Count;
            int diferencia2 = listaPromediosCortoPlazo.Count -
            listaPromediosLargoPlazo.Count;
//Halla la ganacia de cada transaccion en cada fecha y las suma en la
//variable ganancia
            for (int i = 0; i < listaPatronesDeCompra.Count; i++)
            {
//estructura dia para ser utilizada como buffer
                  PantallaPrincipal.dia dia = new
                  PantallaPrincipal.dia();
//Toma la fecha en la cual se identifica el patrón
                  fechaCompra =
                  DateTime.Parse(listaPatronesDeCompra[i].ToString());
                  fechaVenta = DateTime.Parse("01/01/2100");
```

```
//Toma la fecha en la cual se compra la acción que sería al día
//siguiente de identificar el patrón
                  for (int j = 0; j < listaPromediosLargoPlazo.Count -
                  1; j++)
                  {
                        dia = (PantallaPrincipal.dia)listaPrecios[j +
                        diferencia1];
                        if (dia.fecha == fechaCompra)
                        {
                            dia =
                            (PantallaPrincipal.dia)listaPrecios[j + 1
                            + diferencia1];
                            fechaCompra = dia.fecha;
//sale del ciclo al encontrar la fecha de compra
                            break;
                        }
                  }
//Toma el valor inicial de la compra, como si fuera el precio de
//cierre del día siguiente a la identificación del patrón de medias
//móviles
                  valorInicial = dia.valorCierre;
//Si encuentra una fecha de venta mayor que la fecha de compra se toma
//esta como fecha de venta y para el ciclo
                  for (int j = 0; j < listaPatronesDeVenta.Count; j++)
                  {
                   DateTime fechaVentaLocal =
                   DateTime.Parse(listaPatronesDeVenta[j].ToString());
                   if (fechaVentaLocal > fechaCompra)
                   {
                       fechaVenta = fechaVentaLocal;
                       break;
                   }

               }
//Hace la fecha de venta igual a la siguiente de la cual se
//identificó el patrón
               for (int j = 0; j < listaPromediosLargoPlazo.Count - 1;
               j++)
               {
                   dia = (PantallaPrincipal.dia)listaPrecios[j +
                   diferencia1];
                   if (dia.fecha == fechaVenta)
                   {
                      dia = (PantallaPrincipal.dia)listaPrecios[j + 1 +
                      diferencia1];
                      fechaVenta = dia.fecha;
//sale del ciclo al encontrar la fecha de venta
                      break;
                   }
               }

               if (fechaVenta == fechaCompra)
                     fechaVenta = DateTime.Parse("01/01/2100");

//Hace que el valor incial y el final se les quite un 0.5% de su valor
//que sería la comision que se debe pagar al comisionista por la
//transacción
               valorFinal = dia.valorCierre;
               valorInicial = valorInicial * 1.005;
               valorFinal = valorFinal * 0.995;
```

```
//toma el total de las compras para luego restarle las
//inversiones abiertas al final y encontrar así las
//inversiones cerradas
                numeroInversionesCerradas++;
//Aquí decide si la ganacia se toma en el caso de que se haya cerrado
//la transacción en la fecha del día de hoy
                dia =
                (PantallaPrincipal.dia)listaPrecios[listaPrecios.Count -
                1];
                if (dia.fecha > fechaVenta)
                {
                    if (valorInicial < valorFinal)
                    {
                          ganancia += (valorFinal / valorInicial) - 1;
                    }
                    else
                    {
                          perdida += 1 - (valorFinal / valorInicial);
                    }
//Añade las ganancias al reporte
                    double gananciaIndividual = (valorFinal /
                    valorInicial) - 1;
                    anadirAlReporte(this.Text, "cerrada",
                    fechaCompra.ToString(), fechaVenta.ToString(),
                    gananciaIndividual.ToString());

                }
//Si no se ha cerrado la transaccion entonces se almacena en la
//variable que cuenta las transacciones no cerradas, y se decrementa
//la variable que tiene las acciones cerradas
                else
                {
                    PantallaPrincipal.dia buffer =
                    (PantallaPrincipal.dia)listaPrecios[listaPrecios.Cou
                    nt - 1];
                    float gananciaIndividual = (buffer.valorCierre /
                    float.Parse(valorInicial.ToString())) - 1;
                    anadirAlReporte(this.Text, "abierta",
                    fechaCompra.ToString(), "no vendida",
                    gananciaIndividual.ToString());
                    numeroInversionesAbiertas++;
                    if (numeroInversionesCerradas > 0)
                          numeroInversionesCerradas--;
                }
            }
//Retorna el valor de la ganancia como un float
            gananciaGlobal = ganancia;
            perdidaGlobal = perdida;
            ganancia = ganancia - perdida;
//Si la fecha de venta es la de hoy se almacena la fecha
//de compra de la acción para indicar cual es la que se
//debe vender hoy
            if (fechaVenta >= fechaDeHoy && fechaVenta !=
            DateTime.Parse("01/01/2100") && fechaVenta > fechaCompra)
            {
               fechaDeCompraParaLaVentaHoy = fechaCompra.ToString();
            }
//Si la fecha de compra es hoy se hace a la variable
//positiva para indicar que hoy se debe comprar la acción
            if (fechaCompra >= fechaDeHoy)
            {
```

```
                debeInvertir = "SI";
            }
//retorna la ganancia
            return float.Parse(ganancia.ToString());
        }
        return 0;
}
```

CAPITULO 7: OPTIMIZACION DE PERIODOS DE 2 MEDIAS MOVILES EXPONENCIALES PARA HALLAZGO DE PATRONES DE COMPRA Y VENTA CON FILTRO DE VOLUMEN

En el capitulo 6 se desarrolló la optimización de los periodos de 2 medias móviles para que por medio del método del doble cruce se hallaran patrones de compra y de venta, pero algunos de estos patrones pueden no tener un volumen lo suficientemente alto como para que sea un cambio de tendencia fuerte, por lo tanto se hace necesario filtrar dichos patrones con el volumen para tener señales de compra y de venta mas fuertes.

En este caso la optimización consiste en encontrar la mejor combinación de los tres períodos: el período de la media móvil de largo plazo, el de la media móvil de corto plazo y el período del volumen que haya tenido mayor ganancia en la historia de la acción para seguirla utilizando en el futuro.

En la figura 7 se puede ver los resultados del análisis que son: la ganancia bruta, la pérdida bruta, la ganancia total neta, el número de inversiones cerradas y abiertas, los períodos de las medias móviles y el período, así como la conclusión del análisis de si se debe invertir al día siguiente o vender.

El siguiente es el código fuente de las funciones necesarias para esta optimización.

```
//Función para optimizar el período de las medias móviles y el volumen
public void optimizarPeridoConVolumen()
{
      listaReportes = new ArrayList();
      mejorListaReportes = new ArrayList();

//Listas que almacenan los precios y los 2 promedios exponenciales de
//la acción y los precios diarios de la acción
      ArrayList listaPrecios = new ArrayList();
      listaPrecios = listaPreciosGlobal;
      ArrayList listaPromedios1 = new ArrayList();
      ArrayList listaPromedios2 = new ArrayList();
      ArrayList listaPromediosVolumen = new ArrayList();
//Variables que almacenan el mejor período del análisis y el número de
//operaciones abiertas y cerradas, además de la mejor ganancia y las
//conclusiones del análisis
      int mejorPeriodo1 = 1;
```

```
      int mejorPeriodo2 = 1;
      int mejorPeriodoVolumen = 1;
      float menorPerdida = 1000000;
      double mayorGanancia = 0;
      int inversionesAbiertas = 0;
      int inversionesCerradas = 0;
      double gananciaBruta = 0;
      double perdidaBruta = 0;
      string fechaCompraParaLaVenta = "";
      string debeInvertirHoy = "";
//Por cada período de 1 a cien halla el de largo plazo que tenga la
//mejor ganacia, las inversiones abiertas y cerradas
      pbProgreso.Maximum = 100;
      pbProgreso.Minimum = 0;
//Crea un objeto de la clase GraficaVolumen para hallar los volumenes
      GraficaVolumen filtrarConVolumen = new GraficaVolumen();
      for (int i = 1; i < 100; i++)
      {
            listaPromedios1 =
            obtenerMediaMovilExponencial(i.ToString());
//Por cada promedio de corto plazo halla el que tenga mejor ganancia
            for (int j = 1; j < i; j++)
            {
                  listaPromedios2 =
                  obtenerMediaMovilExponencial(j.ToString());
//Por cada promedio de volumen halla el que tenga mejor ganancia
                  for (int k = 1; k < 100; k++)
                  {
                        filtrarConVolumen.listaPreciosGlobal =
                        listaPreciosGlobal;
                        filtrarConVolumen.Text = this.Text;
                        listaPromediosVolumen =
                        filtrarConVolumen.obtenerMediaMovilExponencial
                        (k.ToString());
                        float bufferGanancia =
                        hallarGananciaConVolumen(listaPrecios,
                        listaPromedios1, listaPromedios2,
                        listaPromediosVolumen);
                        if (bufferGanancia > mayorGanancia)
                        {
                              mayorGanancia = bufferGanancia;
                              mejorPeriodo1 = i;
                              mejorPeriodo2 = j;
                              mejorPeriodoVolumen = k;
                              inversionesAbiertas =
                              numeroInversionesAbiertas;
                              inversionesCerradas =
                              numeroInversionesCerradas;
                              gananciaBruta = gananciaGlobal;
                              perdidaBruta = perdidaGlobal;
                              fechaCompraParaLaVenta =
                              fechaDeCompraParaLaVentaHoy;
                              debeInvertirHoy = debeInvertir;
                              mejorListaReportes = listaReportes;
                        }
                  }
            }
            pbProgreso.Value = i;
      }
//Muestra los datos en el formulario
      lbPeriodoGanancias1.Text = mejorPeriodo1.ToString();
```

```
        lbPeriodoGanancias2.Text = mejorPeriodo2.ToString();
        lbPeriodoVolumen.Text = mejorPeriodoVolumen.ToString();

//multiplica por cien para mostrar la ganacia como porcentaje
        menorPerdida *= 100;
//Añande el signo de porcentaje para el valor
        gananciaBruta *= 100;
        perdidaBruta *= 100;
        double ganancia = gananciaBruta - perdidaBruta;
        lbGanancias.Text = ganancia.ToString() + "%";
//Muestra el número de inversiones abiertas y cerradas hasta el
//momento
        lbInversionesAbiertas.Text = inversionesAbiertas.ToString();
        lbInversionesCerradas.Text = inversionesCerradas.ToString();
//Mueatra la ganacia y la perdida brutas en porcentaje y las
//conclusiones del análisis
        lbGanaciaBruta.Text =
        float.Parse(gananciaBruta.ToString()).ToString() + "%";
        lbPerdidaBruta.Text =
        float.Parse(perdidaBruta.ToString()).ToString() + "%";
        lbDebeInvertir.Text = debeInvertirHoy;
        lbFechaCompra.Text = fechaCompraParaLaVenta;
        try
        {
             PantallaPrincipal.dia hoy =
             (PantallaPrincipal.dia)listaPrecios[listaPrecios.Count -
             1];
             fechaDeHoy = hoy.fecha;
             precioDeHoy = hoy.valorCierre;
             PantallaPrincipal.dia ayer =
             (PantallaPrincipal.dia)listaPrecios[listaPrecios.Count -
             2];
             precioDeAyer = ayer.valorCierre;
        }
        catch
        {
        }
}



//Función que halla la ganancia
private float hallarGananciaConVolumen(ArrayList listaPrecios,
ArrayList listaPromediosLargoPlazo, ArrayList
listaPromediosCortoPlazo, ArrayList listaPromediosVolumen)
{
//Función que halla los patrones de medias móviles
      hallarPatronesDeMediasMoviles(listaPrecios,
      listaPromediosLargoPlazo, listaPromediosCortoPlazo);
//filtra los patrones con el vólumen
      filtrarFechasAlcistasConVolumen(listaPatronesDeCompra,
      listaPrecios, listaPromediosVolumen);
      filtrarFechasBajistasConVolumen(listaPatronesDeVenta,
      listaPrecios, listaPromediosVolumen);
//verifica que las listas tengan por lo menos 2 elementos
      if (listaPrecios.Count > 2 && listaPromediosLargoPlazo.Count > 2
      && listaPromediosCortoPlazo.Count > 2)
      {
            listaReportes = new ArrayList();
//variables que almacenan la información de la transacción histórica
            double ganancia = 0;
```

```
            double perdida = 0;
            double valorInicial = 0;
            double valorFinal = 0;
            gananciaGlobal = 0;
            perdidaGlobal = 0;
           DateTime fechaCompra = new DateTime();
           DateTime fechaVenta = new DateTime();
//Variables que almacenan el número de inversiones cerradas y abiertas
//que quedan al final del análisis
           numeroInversionesCerradas = 0;
           numeroInversionesAbiertas = 0;
//Variable que almacena la fecha de compra de la acción que se debe
//vender
           fechaDeCompraParaLaVentaHoy = "";
//Variable que almacena la conclusión de si se debe comprar o no
           debeInvertir = "NO";
//Se inicializan los promedios y el día con el primer elemento de la
//lista
           int diferencia1 = listaPrecios.Count -
           listaPromediosLargoPlazo.Count;
           int diferencia2 = listaPromediosCortoPlazo.Count -
           listaPromediosLargoPlazo.Count;
//Halla la ganacia de cada transacción en cada fecha y las suma en la
//variable ganancia
           for (int i = 0; i < listaPatronesDeCompra.Count; i++)
           {
//estructura día para ser utilizada como buffer
                PantallaPrincipal.dia dia = new
                PantallaPrincipal.dia();
//Toma la fecha en la cual se identifica el patron
                fechaCompra =
                DateTime.Parse(listaPatronesDeCompra[i].ToString());
                fechaVenta = DateTime.Parse("01/01/2100");
//Toma la fecha en la cual se compra la acción que sería al día
//siguiente de identificar el patrón
                for (int j = 0; j < listaPromediosLargoPlazo.Count -
                1; j++)
                {
                      dia = (PantallaPrincipal.dia)listaPrecios[j +
                      diferencia1];
                      if (dia.fecha == fechaCompra)
                      {
                            dia =
                            (PantallaPrincipal.dia)listaPrecios[j +
                            1 + diferencia1];
                            fechaCompra = dia.fecha;
//sale del ciclo al encontrar la fecha de compra
                            break;
                      }
                }

//Toma el valor inicial de la compra, como si fuera el precio de
//cierre del día siguiente a la identificación del patrón de velas
                valorInicial = dia.valorCierre;
//Si encuentra una fecha de venta mayor que la fecha de compra se toma
//esta como fecha de venta y para el ciclo
                for (int j = 0; j < listaPatronesDeVenta.Count; j++)
                {
                      DateTime fechaVentaLocal =
                      DateTime.Parse(listaPatronesDeVenta[j].ToStrin
                      g());
```

```
                        if (fechaVentaLocal > fechaCompra)
                        {
                            fechaVenta = fechaVentaLocal;
                            break;
                        }
                    }
//Hace la fecha de venta igual a la siguiente de la cual se
//identificó el patrón
                    for (int j = 0; j < listaPromediosLargoPlazo.Count -
                    1; j++)
                    {
                        dia = (PantallaPrincipal.dia)listaPrecios[j +
                        diferencia1];
                        if (dia.fecha == fechaVenta)
                        {
                            dia =
                            (PantallaPrincipal.dia)listaPrecios[j +
                            1 + diferencia1];
                            fechaVenta = dia.fecha;
//sale del ciclo al encontrar la fecha de venta
                            break;
                        }
                    }

                    if (fechaVenta == fechaCompra)
                        fechaVenta = DateTime.Parse("01/01/2100");
//Hace que el valor incial y el final se les quite un 0.5% de su valor
//que sería la comisión que se debe pagar al comisionista por la
//transacción
                        valorFinal = dia.valorCierre;
                        valorInicial = valorInicial * 1.005;
                        valorFinal = valorFinal * 0.995;
//toma el total de las compras para luego restarle las
//inversiones abiertas al final y encontrar así las
//inversiones cerradas
                        numeroInversionesCerradas++;
//Aquí decide si la ganacia se toma en el caso de que se haya cerrado
//la transacción en la fecha del día de hoy
                        dia =
                        (PantallaPrincipal.dia)listaPrecios[listaPrecios.
                        Count - 1];
                        if (dia.fecha > fechaVenta)
                        {
                            if (valorInicial < valorFinal)
                            {
                              ganancia += (valorFinal / valorInicial) - 1;
                            }
                            else
                            {
                               perdida += 1 - (valorFinal / valorInicial);
                            }
//Añade las ganancias al reporte
                            double gananciaIndividual = (valorFinal /
                            valorInicial) - 1;
                            anadirAlReporte(this.Text, "cerrada",
                            fechaCompra.ToString(), fechaVenta.ToString(),
                            gananciaIndividual.ToString());

                        }
```

```
//Si no se ha cerrado la transacción, entonces se almacena en la
//variable que cuenta las transacciones no cerradas, y se decrementa
//la variable que tiene las acciones cerradas
                    else
                    {
                        PantallaPrincipal.dia buffer =
                        (PantallaPrincipal.dia)listaPrecios[listaPreci
                        os.Count - 1];
                        float gananciaIndividual = (buffer.valorCierre
                        / float.Parse(valorInicial.ToString())) - 1;
                        anadirAlReporte(this.Text, "abierta",
                        fechaCompra.ToString(), "no vendida",
                        gananciaIndividual.ToString());
                        numeroInversionesAbiertas++;
                        if (numeroInversionesCerradas > 0)
                            numeroInversionesCerradas--;
                    }
                }

//Retorna el valor de la pérdida como un float
                gananciaGlobal = ganancia;
                perdidaGlobal = perdida;
                ganancia = ganancia - perdida;

//Si la fecha de venta es la de hoy se almacena la fecha
//de compra de la acción para indicar cual es la que se
//debe vender mañana
            if (fechaVenta >= fechaDeHoy && fechaVenta !=
            DateTime.Parse("01/01/2100") && fechaVenta > fechaCompra)
            {
                fechaDeCompraParaLaVentaHoy = fechaCompra.ToString();
            }
//Si la fecha de compra es hoy se hace a la variable
//positiva para indicar que hoy se debe comprar la acción
            if (fechaCompra >= fechaDeHoy)
            {
                debeInvertir = "SI";
            }
//retorna la ganancia
            return float.Parse(ganancia.ToString());
        }
        return 0;
}



//Función para filtrar las fechas bajistas con volumen
private ArrayList filtrarFechasBajistasConVolumen(ArrayList
listaPatronesReversaBajistasLocal, ArrayList listaPrecios, ArrayList
listaPromedios)
{
//Filtra las fechas en los patrones de reversa bajistas que no tengan
//el volumen suficiente
//Encuentra cual lista tiene menor tamaño para luego empezar a leer
//desde el elemento correcto

        int minimoTamaño = 0;
        if (listaPrecios.Count < listaPromedios.Count)
        {
            minimoTamaño = listaPrecios.Count;
        }
```

```
      else
      {
          minimoTamaño = listaPromedios.Count;
      }
//halla la diferencia entre los tamaños de las listas
      for (int i = 0; i < minimoTamaño; i++)
      {
          int diferencia = 0;
//Si la lista de promedios es menor que la de precios
//se halla la diferencia y se toma el día de lista precios
//que concuerda con el primero de la lista de promedios
          if (listaPromedios.Count < listaPrecios.Count)
          {
             diferencia = listaPrecios.Count - listaPromedios.Count;
             PantallaPrincipal.dia dia =
            (PantallaPrincipal.dia)listaPrecios[i + diferencia];
//Se filtra la fecha si no cumple con el volumen necesario
//que es el uno por ciento mayor que el de la media
             float promedio = (float)listaPromedios[i];
             if (dia.volumen < promedio * 1.01)
             {
                  if
                  (listaPatronesReversaBajistasLocal.Contains(dia.fech
                  a))
                  listaPatronesReversaBajistasLocal.Remove(dia.fecha);
             }
        }
//Si la lista de promedios es mayor que la de precios
//se halla la diferencia y se toma el día de lista promedios
//que concuerda con el primero de la lista de precios
        else
        {
            diferencia = listaPromedios.Count - listaPrecios.Count;
            PantallaPrincipal.dia dia =
            (PantallaPrincipal.dia)listaPrecios[i];
//Se filtra la fecha si no cumple con el volumen necesario
//que es el uno por ciento mayor que el de la media
            float promedio = (float)listaPromedios[i + diferencia];
            if (dia.volumen < promedio * 1.01)
            {
                  if
                  (listaPatronesReversaBajistasLocal.Contains(dia.fech
                  a))
                  listaPatronesReversaBajistasLocal.Remove(dia.fecha);
            }
      }
  }
//Retorna las fechas de patrones bajistas filtrados
  return listaPatronesReversaBajistasLocal;
}



//Función para filtrar las fechas alcistas con volumen
private ArrayList filtrarFechasAlcistasConVolumen(ArrayList
listaPatronesReversaAlcistasLocal, ArrayList listaPrecios, ArrayList
listaPromedios)
{
//Filtra las fechas en los patrones de reversa alcistas que no tengan
//el volumen suficiente
```

```
//Encuentra cual lista tiene menor tamaño para luego empezar a leer
//desde el elemento correcto
    int minimoTamaño = 0;
    if (listaPrecios.Count < listaPromedios.Count)
    {
         minimoTamaño = listaPrecios.Count;
    }
    else
    {
      minimoTamaño = listaPromedios.Count;
   }

//halla ladiferencia entre los tamaños de las listas
   for (int i = 0; i < minimoTamaño; i++)
   {
       int diferencia = 0;

//Si la lista de promedios es menor que la de precios
//se halla la diferencia y se toma el día de lista precios
//que concuerda con el primero de la lista de promedios
       if (listaPromedios.Count < listaPrecios.Count)
       {
            diferencia = listaPrecios.Count - listaPromedios.Count;
            PantallaPrincipal.dia dia =
            (PantallaPrincipal.dia)listaPrecios[i + diferencia];
//Se filtra la fecha si no cumple con el volumen necesario
//que es el uno por ciento mayor que el de la media
            float promedio = (float)listaPromedios[i];
            if (dia.volumen < promedio * 1.01)
            {
                  if
                  (listaPatronesReversaAlcistasLocal.Contains(dia.fech
                  a))
                  listaPatronesReversaAlcistasLocal.Remove(dia.fecha);
            }
        }
//Si la lista de promedios es mayor que la de precios
//se halla la diferencia y se toma el día de lista promedios
//que concuerda con el primero de la lista de precios
        else
        {
             diferencia = listaPromedios.Count - listaPrecios.Count;
             PantallaPrincipal.dia dia =
            (PantallaPrincipal.dia)listaPrecios[i];
//Se filtra la fecha si no cumple con el volumen necesario
//que es el uno por ciento mayor que el de la media
             float promedio = (float)listaPromedios[i + diferencia];
             if (dia.volumen < promedio * 1.01)
             {
                  if
                  (listaPatronesReversaAlcistasLocal.Contains(dia.fech
                  a))
                  listaPatronesReversaAlcistasLocal.Remove(dia.fecha);
             }
         }
     }
//Retorna la lista de fechas de patrones alcistas filtrados
     return listaPatronesReversaAlcistasLocal;
    }
  }
}
```

BIBLIOGRAFIA

Alfredo Díaz Mata, Los Osciladores %K y %R del análisis técnico bursátil y una propuesta para mejorar el %R.

Antonio de la Torre Gallegos, Utilidad de los nuevos indicadores técnicos en el análisis bursátil.

Bodie, Kane y Marcus (2004), Principios de Inversiones.

Francisco Javier Ceballos, Enciclopedia de Microsoft Visual C#.

Miguel Rodríguez Gómez Stern y Marco Antonio Besteiro Gorostizaga, Desarrollo de aplicaciones .NET con visual C#.

MORPHY J.J, (2000), “Análisis Técnico de los Mercados Financieros”.

Omar y Jimmy Hernández, Como Hacerse Millonario en la Bolsa de Valores.