1

Дослідження моделей та засобів для аналізу та прогнозування валютних котувань

Виконав: ст.гр.ІПЗм-17-2

Чушенко Сергій

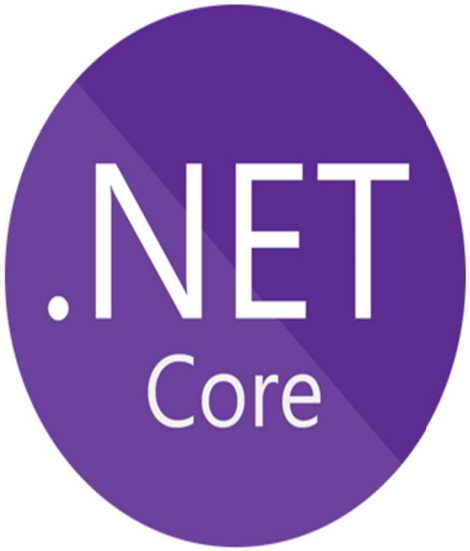
Керівник роботи: проф. Єрохін А.Л.

Постановка задачі

- Долідження та аналіз ринку валютних котувань.
- Дослідження основних моделей та методів прогнозу валют.
- Порівняння методів і моделей для вибору найбільш точного алгоритму.
- Моделювання та проектування програмної системи.
- Розробка програмної системи.
- Написання пояснювальної записки.

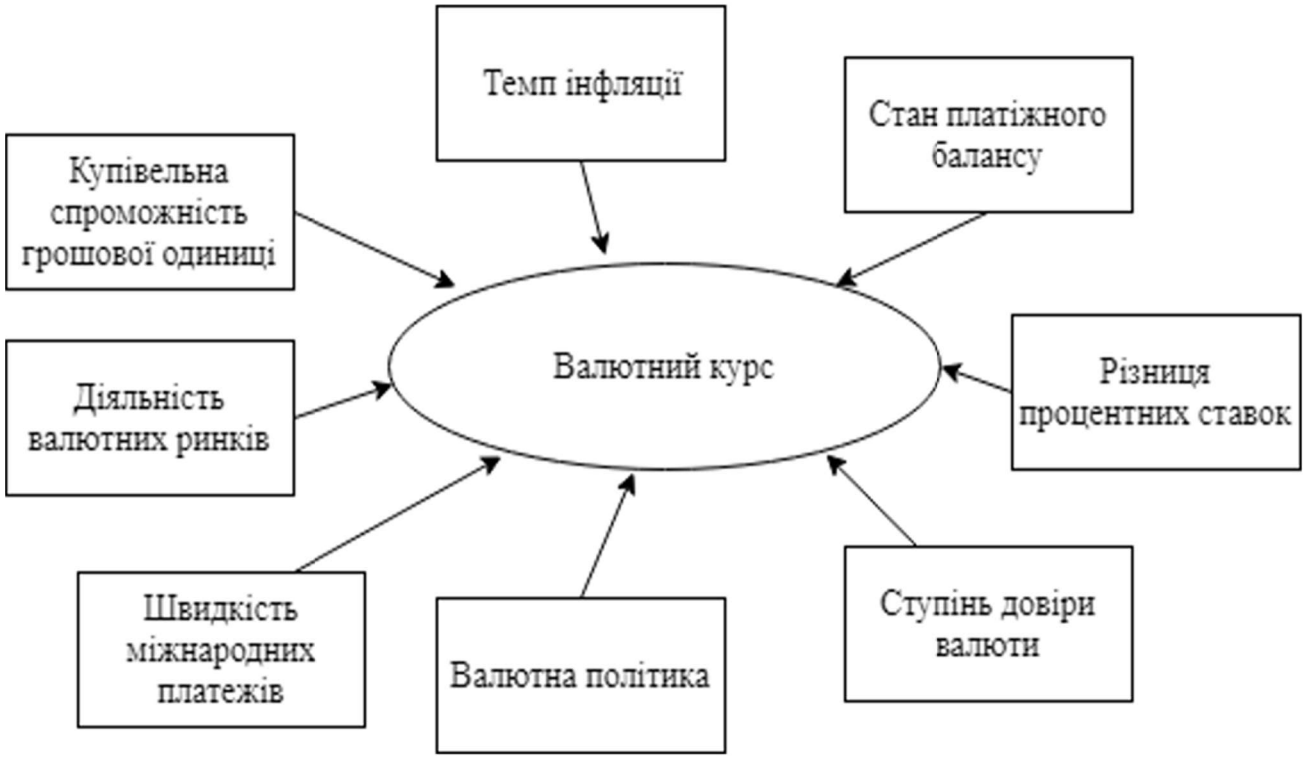
3

Технології



Аналіз економічних факторів, що впливають на курси валют

4



Основні поняття

$$E_n = C_f / C_d$$

де E_n – номінальний валютний курс;

C_f – іноземна валюта;

C_d – національна валюта.

$$E_r = E_n \times (P_f / P_d)$$

де E_r – реальний валютний курс;

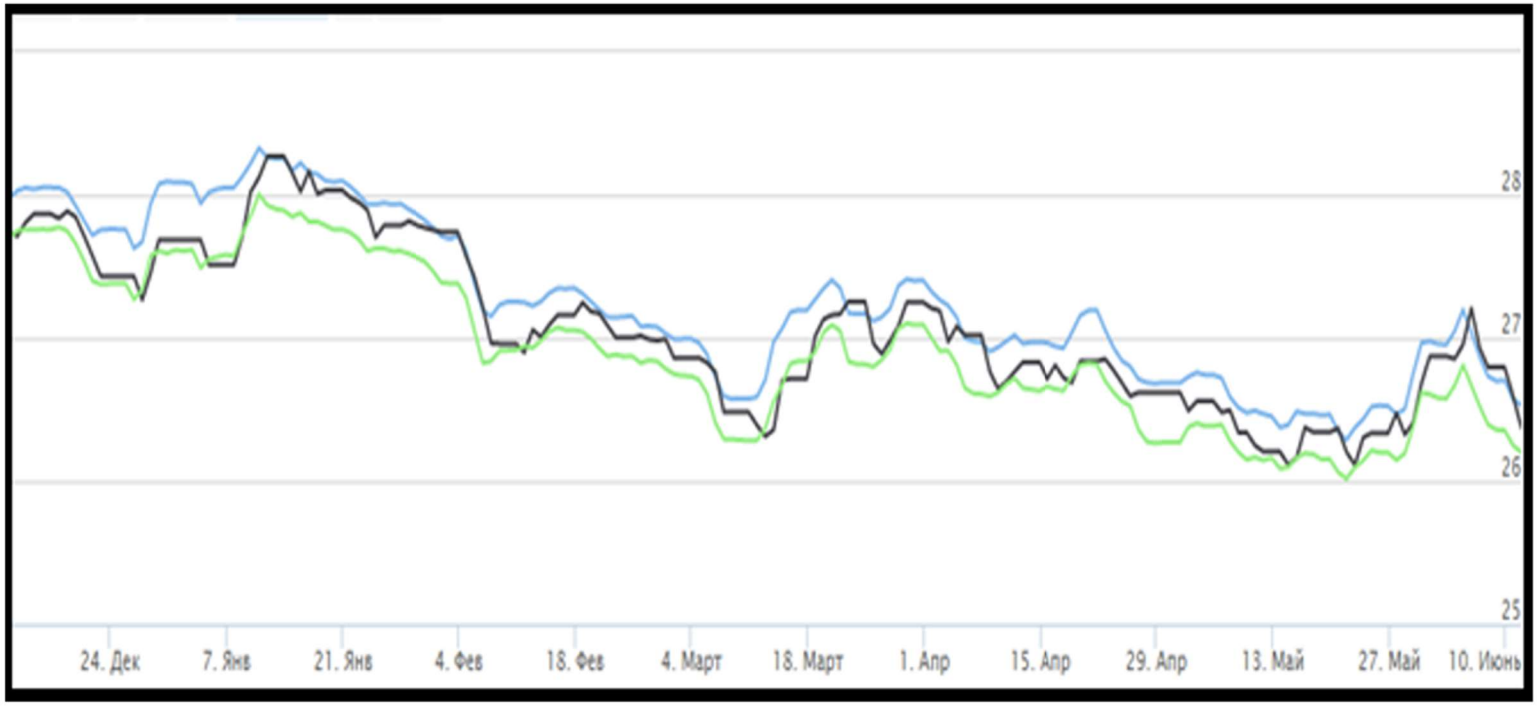
E_n – номінальний валютний курс;

P_f – індекс цін зарубіжної країни;

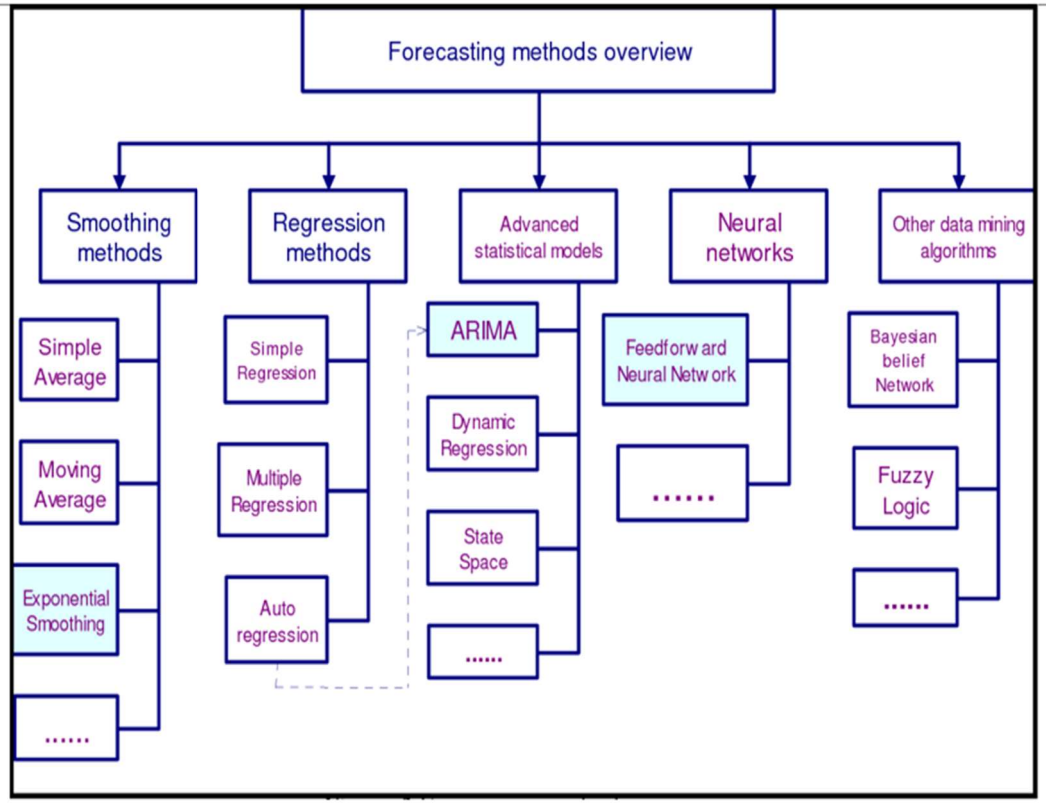
P_d – індекс цін своєї країни.

6

Аналіз курсу валют

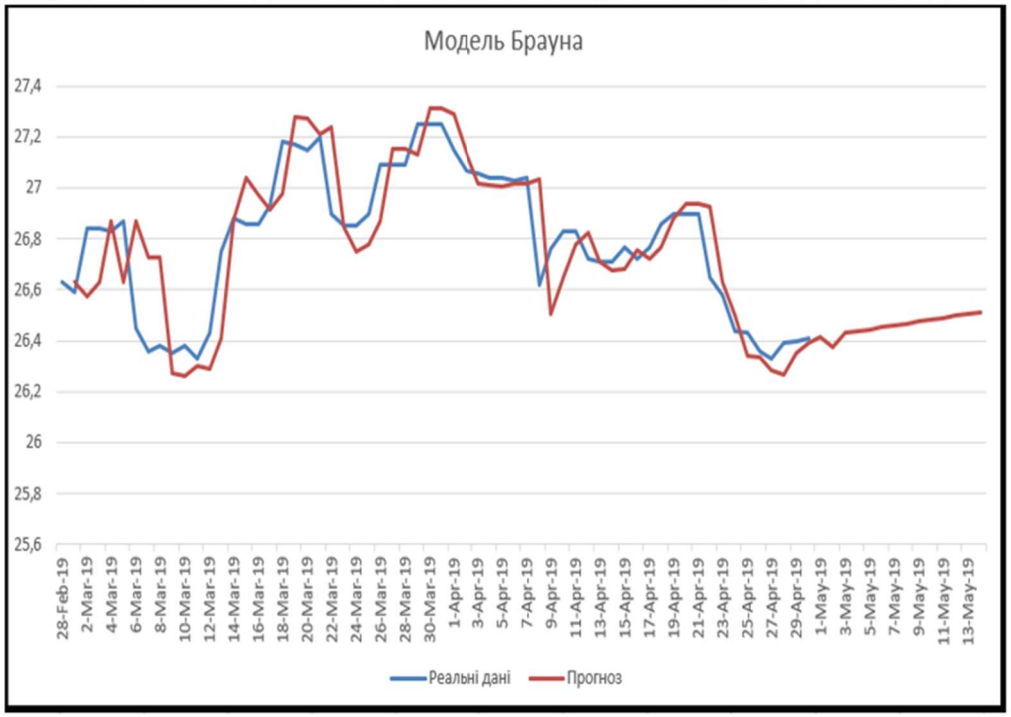


Аналіз методів прогнозування



8

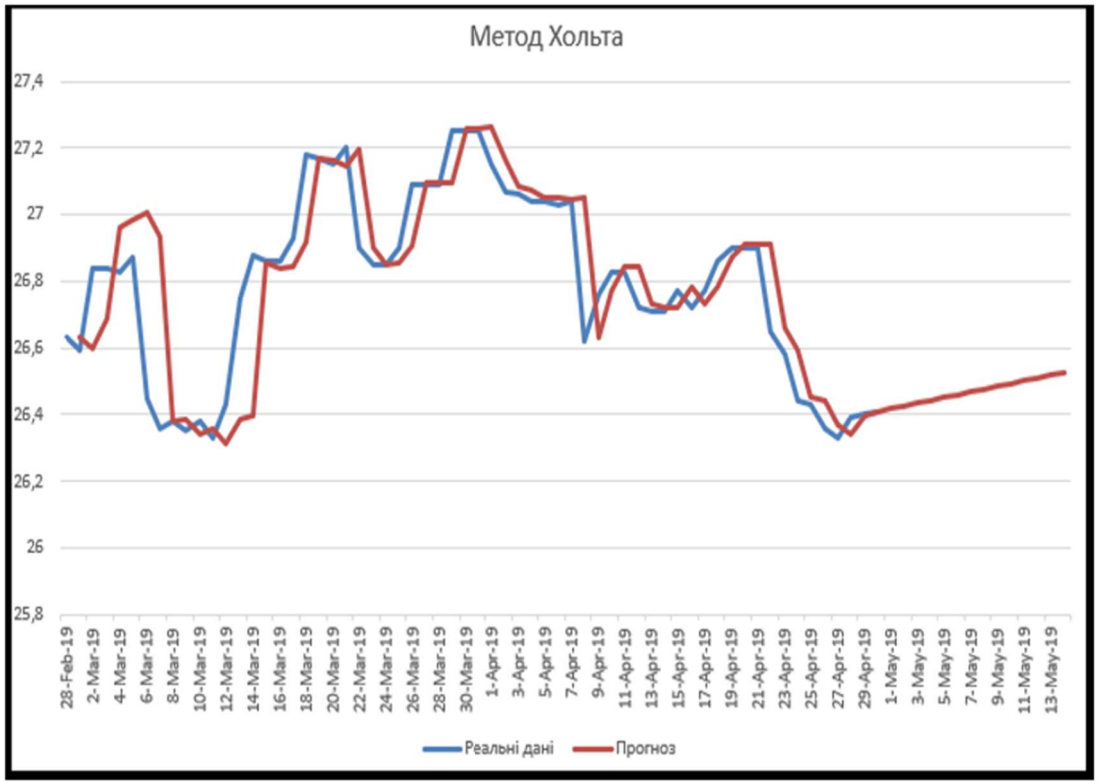
Модель Брауна



$$y_{th}(\tau) = a_{0(1)} + a_{1(t)}\tau$$

$$y_{th}(n+\tau) = a_{0(n)} + a_{1(n)}\tau, \tau = 1,2....$$

Метод Хольта



$$T_t = b*(L_t - L_{t-1}) + (1-b)*T_{t-1}$$

де T_t – значення тренду за поточний період;

b – коефіцієнт згладжування тренду;

L_t – експоненціально згладжена величина за поточний період;

L_{t-1} – експоненціально згладжена величина за попередній період;

T_{t-1} – значення тренду за попередній період.

$$Y_{t+p} = Lt + p*Tt$$

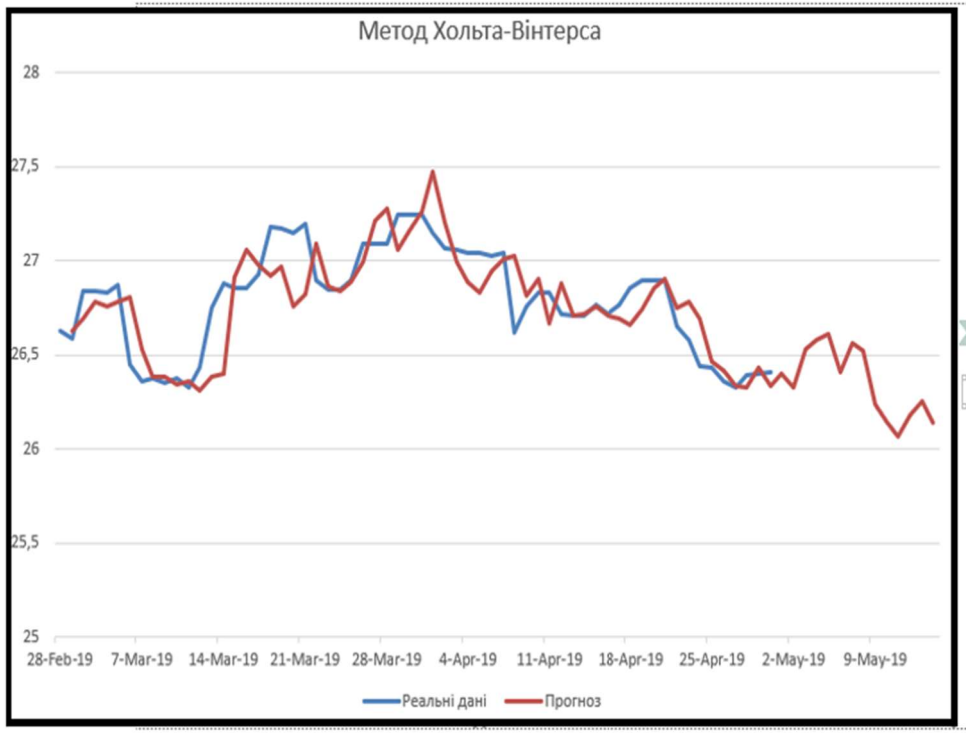
де Y_{t+p} – прогноз по методу Хольта на p період;

L_t – експоненціально згладжена величина за останній період;

p – порядковий номер періоду, на який робимо прогноз;

T_t – тренд за останній період.

Метод Хольта-Вінтерса



$$S_t = q * Y_t / L_t + (1 - q) * S_{t-s}$$

де S_t – коефіцієнт сезонності для поточного періоду;

q – коефіцієнт згладжування сезонності;

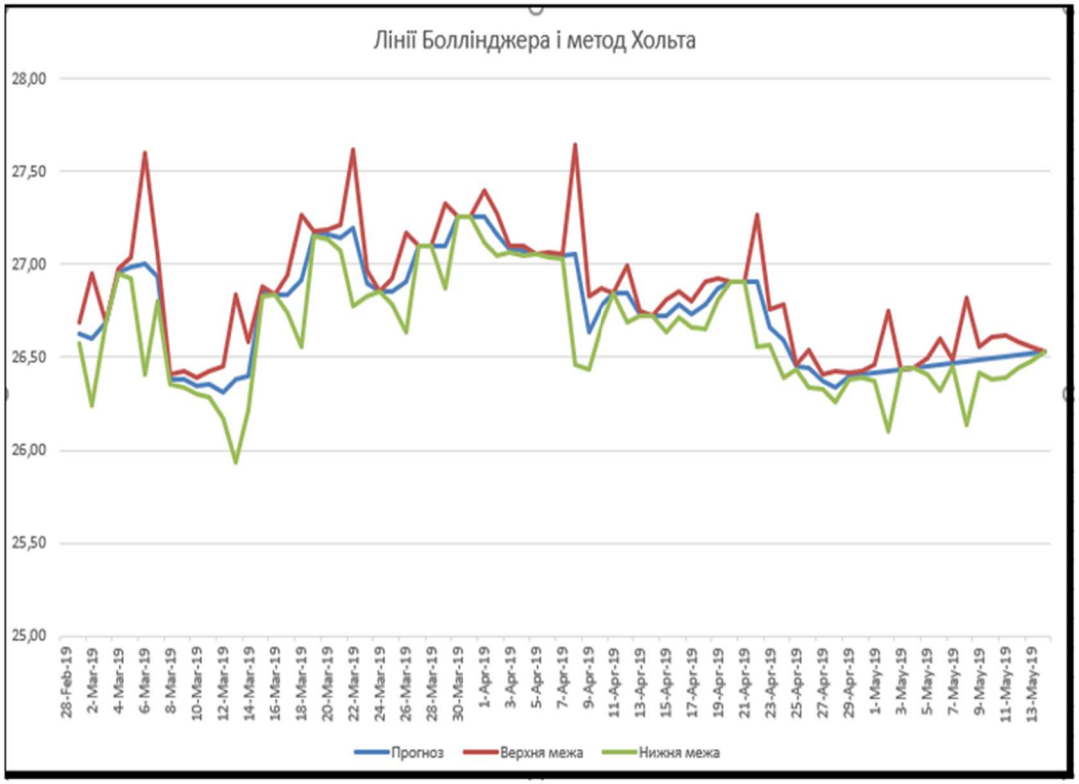
Y_t – поточне значення ряду (наприклад, обсяг продажу);

L_t – згладжена величина за поточний період;

S_{t-s} – коефіцієнт сезонності за цей же період в попередньому сезоні.

$$Y_{t+p} = (L_t + p * T_t) * S_{t-s+p}$$

Метод Хольта+лінії Болліджера



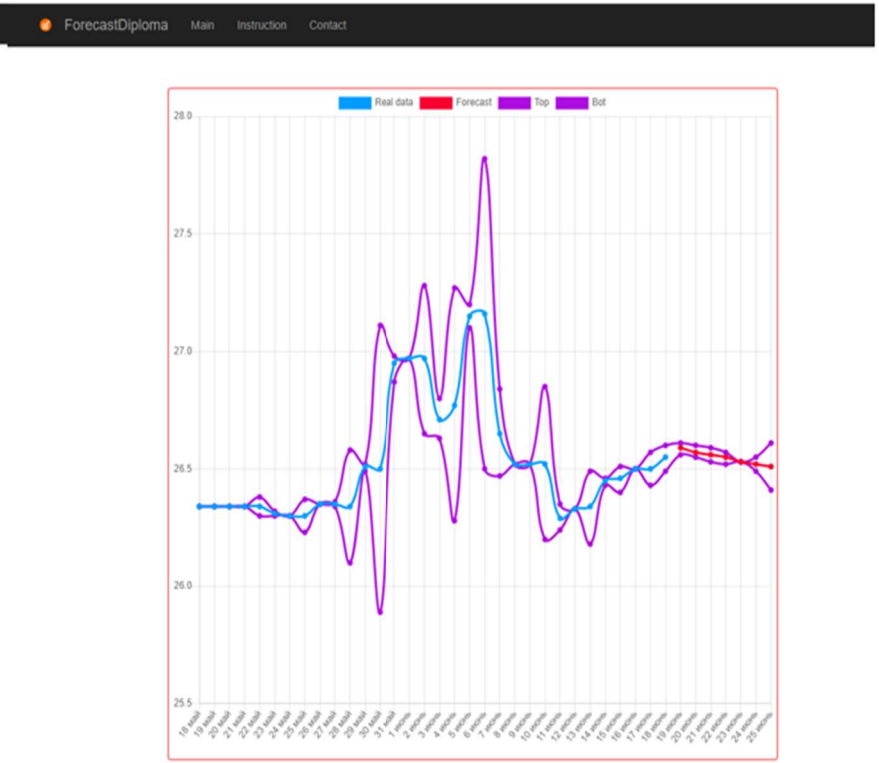
$$TL = ML \pm (D * StdDev)$$

де TL – верхня або нижня лінія Болліджера;

D – кількість стандартних відхилень;

$StdDev$ – значення стандартного відхилення.

Реалізація програмного продукту



Висновки

- Комбінований метод Хольта з лініями Болліджера показав хороші результати, з низькою середньоквадратичною помилкою.

- Курс валют досить таки складно прогнозувати, це пов'язано з тим, що на нього впливають велика кількість економічних факторів.

- Була спроектована, реалізована і протестована програмна система.

- Проаналізовано основні методи прогнозування часових рядів.

14

Дякую за увагу!

ДОДАТОК Б

Лістинг коду програми

```
@{
    ViewBag.Title = "Home Page";
}
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <script src="~/Scripts/Chart.js"></script>
    <script src="~/Scripts/jquery-3.3.1.js"></script>
    <script src="~/Scripts/bootstrap.min.js"></script>
    <link href="~/Content/bootstrap.min.css" rel="stylesheet" />
    <style>
        #chart_container{
            width:800px;
            height:800px;
            border:1px solid #ff0000;
            padding:1px;
            border-radius:4px;
            margin:0 auto;
            margin-top:50px
        }
    </style>
</head>
<body>
    <div id="chart_container" >
```

```
    <canvas id="bar_chart"></canvas>
</div>
<script>

    var ages = @Html.Raw(Json.Encode(ViewBag.Dates));
    var repos = @Html.Raw(Json.Encode(ViewBag.AGES));
    var prog = @Html.Raw(Json.Encode(ViewBag.Prog));
    var legs = @Html.Raw(Json.Encode(ViewBag.Legs));
    var bot = @Html.Raw(Json.Encode(ViewBag.Bot));
    var ctx = $("#bar_chart");
    var barChart = new Chart(ctx, {
        type: 'line',
        data: {
            labels: ages,
            datasets: [{
                label: "Real data",
                data: repos,
                backgroundColor: 'rgba(0, 157, 255, 1)',
                borderColor:'rgba(0, 157, 255, 1)',
                fill:false
            }, {
                    label: "Forecast",
                    data: prog,
                    backgroundColor: 'rgba(250, 0, 50, 1)',
                    borderColor: 'rgba(250, 0, 50, 1)',
                    fill: false
```

```
                    }, {
                        label: "Top",
                        data: legs,
                        backgroundColor: 'rgba(175, 13, 224, 1)',
                        borderColor: 'rgba(175, 13, 224, 1)',
                        fill:false
                    },
                    {
                        label: "Bot",
                        data: bot,
                        backgroundColor: 'rgba(175, 13, 224, 1)',
                        borderColor: 'rgba(175, 13, 224, 1)',
                        fill:false
                    }
                ]
            },
            options: {
                maintainAspectRatio: false,
                legend: { display: true }
            }
        });
    </script>
</body>
function __embind_register_smart_ptr(rawType, rawPointeeType, name,
sharingPolicy, getPointeeSignature, rawGetPointee, constructorSignature,
```

```
rawConstructor, shareSignature, rawShare, destructorSignature,
rawDestructor) {
    name = readLatin1String(name);
    rawGetPointee = requireFunction(getPointeeSignature, rawGetPointee);
    rawConstructor = requireFunction(constructorSignature, rawConstructor);
    rawShare = requireFunction(shareSignature, rawShare);
    rawDestructor = requireFunction(destructorSignature, rawDestructor);
    whenDependentTypesAreResolved([rawType], [rawPointeeType],
(function(pointeeType) {
        pointeeType = pointeeType[0];
        var registeredPointer = new RegisteredPointer(name,
pointeeType.registeredClass, false, false, true, pointeeType,
sharingPolicy, rawGetPointee, rawConstructor, rawShare, rawDestructor);
        return [registeredPointer]
    }))
}

function _llvm_exp2_f32(x) {
    return Math.pow(2, x)
}

function _llvm_exp2_f64() {
    return _llvm_exp2_f32.apply(null, arguments)
}
Module["_pthread_cond_broadcast"] = _pthread_cond_broadcast;

function new_(constructor, argumentList) {
    if (!(constructor instanceof Function)) {
        throw new TypeError("new_ called with constructor type " + typeof
constructor + " which is not a function")
    }
    var dummy = createNamedFunction(constructor.name ||
"unknownFunctionName", (function() {}));
    dummy.prototype = constructor.prototype;
    var obj = new dummy;
    var r = constructor.apply(obj, argumentList);
    return r instanceof Object ? r : obj
}

function craftInvokerFunction(humanName, argTypes, classType,
cppInvokerFunc, cppTargetFunc) {
    var argCount = argTypes.length;
    if (argCount < 2) {
        throwBindingError("argTypes array size mismatch! Must at least get
return value and 'this' types!")
    }
    var isClassMethodFunc = argTypes[1] !== null && classType !== null;
    var argsList = "";
    var argsListWired = "";
    for (var i = 0; i < argCount - 2; ++i) {
        argsList += (i !== 0 ? ", " : "") + "arg" + i;
```

```
        argsListWired += (i !== 0 ? ", " : "") + "arg" + i + "Wired"
    }
    var invokerFnBody = "return function " +
makeLegalFunctionName(humanName) + "(" + argsList + ") {\n" + "if
(arguments.length !== " + (argCount - 2) + ") {\n" +
"throwBindingError('function " + humanName + " called with ' +
arguments.length + ' arguments, expected " + (argCount - 2) + " args!');\n"
+ "}\n";
    var needsDestructorStack = false;
    for (var i = 1; i < argTypes.length; ++i) {
        if (argTypes[i] !== null && argTypes[i].destructorFunction ===
undefined) {
            needsDestructorStack = true;
            break
        }
    }
    if (needsDestructorStack) {
        invokerFnBody += "var destructors = [];\n"
    }
    var dtorStack = needsDestructorStack ? "destructors" : "null";
    var args1 = ["throwBindingError", "invoker", "fn", "runDestructors",
"retType", "classParam"];
    var args2 = [throwBindingError, cppInvokerFunc, cppTargetFunc,
runDestructors, argTypes[0], argTypes[1]];
    if (isClassMethodFunc) {
        invokerFnBody += "var thisWired = classParam.toWireType(" +
dtorStack + ", this);\n"
    }
    for (var i = 0; i < argCount - 2; ++i) {
        invokerFnBody += "var arg" + i + "Wired = argType" + i +
".toWireType(" + dtorStack + ", arg" + i + "); // " + argTypes[i + 2].name
+ "\n";
        args1.push("argType" + i);
        args2.push(argTypes[i + 2])
    }
    if (isClassMethodFunc) {
        argsListWired = "thisWired" + (argsListWired.length > 0 ? ", " :
"") + argsListWired
    }
    var returns = argTypes[0].name !== "void";
    invokerFnBody += (returns ? "var rv = " : "") + "invoker(fn" +
(argsListWired.length > 0 ? ", " : "") + argsListWired + ");\n";
    if (needsDestructorStack) {
        invokerFnBody += "runDestructors(destructors);\n"
    } else {
        for (var i = isClassMethodFunc ? 1 : 2; i < argTypes.length; ++i) {
            var paramName = i === 1 ? "thisWired" : "arg" + (i - 2) +
"Wired";
            if (argTypes[i].destructorFunction !== null) {
                invokerFnBody += paramName + "_dtor(" + paramName + "); //
" + argTypes[i].name + "\n";
```

```
                args1.push(paramName + "_dtor");
                args2.push(argTypes[i].destructorFunction)
            }
        }
    }
    if (returns) {
        invokerFnBody += "var ret = retType.fromWireType(rv);\n" + "return
ret;\n"
    } else {}
    invokerFnBody += "}\n";
    args1.push(invokerFnBody);
    var invokerFunction = new_(Function, args1).apply(null, args2);
    return invokerFunction
}

function ensureOverloadTable(proto, methodName, humanName) {
    if (undefined === proto[methodName].overloadTable) {
        var prevFunc = proto[methodName];
        proto[methodName] = (function() {
            if
(!proto[methodName].overloadTable.hasOwnProperty(arguments.length)) {
                throwBindingError("Function '" + humanName + "' called with
an invalid number of arguments (" + arguments.length + ") - expects one of
(" + proto[methodName].overloadTable + ")!")
            }
            return
proto[methodName].overloadTable[arguments.length].apply(this, arguments)
        });
        proto[methodName].overloadTable = [];
        proto[methodName].overloadTable[prevFunc.argCount] = prevFunc
    }
}

function heap32VectorToArray(count, firstElement) {
    var array = [];
    for (var i = 0; i < count; i++) {
        array.push(HEAP32[(firstElement >> 2) + i])
    }
    return array
}
var UnboundTypeError = undefined;

function throwUnboundTypeError(message, types) {
    var unboundTypes = [];
    var seen = {};

    function visit(type) {
        if (seen[type]) {
            return
        }
        if (registeredTypes[type]) {
```

```
            return
        }
        if (typeDependencies[type]) {
            typeDependencies[type].forEach(visit);
            return
        }
        unboundTypes.push(type);
        seen[type] = true
    }
    types.forEach(visit);
    throw new UnboundTypeError(message + ": " +
unboundTypes.map(getTypeName).join([", "]))
}

function __embind_register_class_class_function(rawClassType, methodName,
argCount, rawArgTypesAddr, invokerSignature, rawInvoker, fn) {
    var rawArgTypes = heap32VectorToArray(argCount, rawArgTypesAddr);
    methodName = readLatin1String(methodName);
    rawInvoker = requireFunction(invokerSignature, rawInvoker);
    whenDependentTypesAreResolved([], [rawClassType], (function(classType)
{
        classType = classType[0];
        var humanName = classType.name + "." + methodName;

        function unboundTypesHandler() {
            throwUnboundTypeError("Cannot call " + humanName + " due to
unbound types", rawArgTypes)
        }
        var proto = classType.registeredClass.constructor;
        if (undefined === proto[methodName]) {
            unboundTypesHandler.argCount = argCount - 1;
            proto[methodName] = unboundTypesHandler
        } else {
            ensureOverloadTable(proto, methodName, humanName);
            proto[methodName].overloadTable[argCount - 1] =
unboundTypesHandler
        }
        whenDependentTypesAreResolved([], rawArgTypes, (function(argTypes)
{
            var invokerArgsArray = [argTypes[0],
null].concat(argTypes.slice(1));
            var func = craftInvokerFunction(humanName, invokerArgsArray,
null, rawInvoker, fn);
            if (undefined === proto[methodName].overloadTable) {
                func.argCount = argCount - 1;
                proto[methodName] = func
            } else {
                proto[methodName].overloadTable[argCount - 1] = func
            }
            return []
        }));
```

```
   public static double[] forecast(long[] y, double alpha, double beta,
            double gamma, int period, int m, boolean debug) {
        validateArguments(y, alpha, beta, gamma, period, m);
        int seasons = y.length / period;
        double a0 = calculateInitialLevel(y);
        double b0 = calculateInitialTrend(y, period);
        double[] initialSeasonalIndices = calculateSeasonalIndices(y,
period,seasons);
        if (debug) {
            result+=String.format(
                    "Total observations: %d, Seasons %d, Periods %d",
y.length,seasons, period));
            result+=String.Format("Initial level value a0: " + a0);
            result+=String.Format("Initial trend value b0: " + b0);
            printArray("Seasonal Indices: ", initialSeasonalIndices);
        }
        double[] forecast = calculateHoltWinters(y, a0, b0, alpha, beta,
gamma,initialSeasonalIndices, period, m, debug);
        if (debug) {
            printArray("Forecast", forecast);
        }
        return forecast;
    }
    public static double[] forecast(long[] y, double alpha, double beta,
            double gamma, int period, int m) {
        return forecast(y, alpha, beta, gamma, period, m, false);
    }
```

```
    private static void validateArguments(long[] y, double alpha, double
beta,double gamma, int period, int m) {

        if (y == null) {

            throw new Exception("Value of y should be not null");

        }

        if(m <= 0){

            throw new Exception("Value of m must be greater than 0.");

        }

        if(m > period){

            throw new Exception("Value of m must be <= period.");

        }

        if((alpha < 0.0) || (alpha > 1.0)){

            throw new Exception("Value of Alpha should satisfy 0.0 <= alpha
<= 1.0");

        }

        if((beta < 0.0) || (beta > 1.0)){

            throw new Exception("Value of Beta should satisfy 0.0 <= beta
<= 1.0");

        }

        if((gamma < 0.0) || (gamma > 1.0)){

            throw new Exception("Value of Gamma should satisfy 0.0 <= gamma
<= 1.0");

        }

    }

    private static double[] calculateHoltWinters(long[] y, double a0,
double b0,double alpha, double beta, double gamma,double[]
initialSeasonalIndices, int period, int m, boolean debug) {

        double[] St = new double[y.length];

        double[] Bt = new double[y.length];
```

```
        double[] It = new double[y.length];
        double[] Ft = new double[y.length + m];
        // Initialize base values
        St[1] = a0;
        Bt[1] = b0;
        for (int i = 0; i < period; i++) {
            It[i] = initialSeasonalIndices[i];
        }
        // Start calculations
        for (int i = 2; i < y.length; i++) {
            // Calculate overall smoothing
            if ((i - period) >= 0) {
                St[i] = alpha * y[i] / It[i - period] + (1.0 - alpha)
                        * (St[i - 1] + Bt[i - 1]);
            } else {
                St[i] = alpha * y[i] + (1.0 - alpha) * (St[i - 1] + Bt[i -
1]);
            }
            // Calculate trend smoothing
            Bt[i] = gamma * (St[i] - St[i - 1]) + (1 - gamma) * Bt[i - 1];
            // Calculate seasonal smoothing
            if ((i - period) >= 0) {
                It[i] = beta * y[i] / St[i] + (1.0 - beta) * It[i -
period];
            }

            // Calculate forecast
```

```
            if (((i + m) >= period)) {
                Ft[i + m] = (St[i] + (m * Bt[i])) * It[i - period + m];
            }
            if (debug) {
                result+=String.format(
                        "i = %d, y = %d, S = %f, Bt = %f, It = %f, F = %f",
i,y[i], St[i], Bt[i], It[i], Ft[i]));
            }
        }
        return Ft;
    private static double calculateInitialTrend(long[] y, int period) {
        double sum = 0;
        for (int i = 0; i < period; i++) {
            sum += (y[period + i] - y[i]);
        }
        return sum / (period * period);
    }
    private static double[] calculateSeasonalIndices(long[] y, int period,
            int seasons) {
        double[] seasonalAverage = new double[seasons];
        double[] seasonalIndices = new double[period];
        double[] averagedObservations = new double[y.length];
        for (int i = 0; i < seasons; i++) {
            for (int j = 0; j < period; j++) {
                seasonalAverage[i] += y[(i * period) + j];
            }
```

```
            seasonalAverage[i] /= period;

        }

        for (int i = 0; i < seasons; i++) {

            for (int j = 0; j < period; j++) {

                averagedObservations[(i * period) + j] = y[(i * period) +
j]/ seasonalAverage[i];

            }

        for (int i = 0; i < period; i++) {

            for (int j = 0; j < seasons; j++) {

                seasonalIndices[i] += averagedObservations[(j * period) +
i];

            }

            seasonalIndices[i] /= seasons;

        }

        return seasonalIndices;

    }
```

ДОДАТОК В

Наукові публікації

В.1 Тезіси на конференцію MicroCAD-2019 в ХПІ.

ISSN 2222-2944. Інформаційні технології: наука, техніка, технологія, освіта, здоров'я. 2019. Ч. I

ДОСЛІДЖЕННЯ МОДЕЛЕЙ ТА ЗАСОБІВ ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ВАЛЮТНИХ КОТУВАНЬ

Чушенко С.І., Єрохін А.Л.
Харківський національний університет радіоелектроніки, м. Харків

Робота присвячена розробці ефективної моделі для прогнозування валютних котувань. Передбачення подальшої поведінки фінансових ринків має велике значення для підприємницької діяльності, так як дає колосальні можливості для оптимізації фінансових втрат.

Мета дослідження – на основі аналізу існючого масиву методів прогнозування створити вдосконалений метод та модель прогнозування змінюваності валют, а також побудувати інформаційну систему з графічно. візуалізацією даних. Так як на прогнозування впливають безліч економічних факторів, то довгострокове передбачення практично неможливо. Тому для дослідження було обрано найбільш точний короткостроковий алгоритм. Для прогнозу динаміки будь-якої валютної пари в майбутньому використовується множина розроблених методик. Усі вони засновані на врахуванні факторів, які формують валютний попит і пропозицію на світовому ринку. Тож предметом дослідження є підходи та технічні заходи, що дозволяюсь створити систему прогнозування курсів валют використовуючи специфічні технології та пов'язані з ними стандарти, що уможливлюють взаємозв'язок систем, їх подальший розвиток та вдосконалення.

Хоча прогнозування не є повністю точною наукою та часто прогнози не можна вважати достатньо надійними, однак загальне розуміння того, які економічні сили впливають на формування валютних курсів, дозволяє інвесторам своєчасно реагувати на очікувані зміни майбутньої їх динаміки. Результат офіційних інтервенцій на валютному ринку залежить від реакції спекулянтів. Якщо останні переконані, що центральний банк повинен стабілізувати той чи інший валютний курс, то вони будуть сприяти підтримці курсу цієї валюти.

Суть запропонованого вдосконаленого метода полягає у комплексному врахуванні факторів впливу, які надходять як від ринку, так і цілеспрямованих дій уряду та створення прогнозу на основі модифікованого методу ковзної середньої, що дозволяє застосовувати гнучкий регресійний аналіз. Також в роботі запропонована модель побудови прогнозу та інформаціна система для прогнозування валютних котувань, що складається з наступних модулів: модуль збору даних, знаходження тренду та модуль прогнозування.

Рисунок В.1 – Сторінка 1

ДОДАТОК Г

Електронні матеріали (CD)