ДОДАТОК А

Лістинг коду веб-клієнту

package.json

```
"peerDependencies": {
    "react": "16.3.2",
    "react-dom": "16.3.2",
    "react-intl": "2.4.0"
  },
  "engines": {
    "node": "9.x",
    "npm": "5.8.x"
  },
  "dependencies": {
    "@material-ui/core": "3.9.3",
    "prop-types": "15.6.1",
    "react-redux": "5.0.7",
    "react-router-dom": "4.2.2",
    "redux": "4.0.0",
    "redux-thunk": "2.2.0",
  }
```

actions.js

```
export const calculate = ({ dependent, independents }) => dispatch =>

postJson({
  uri: 'http://localhost:8080/rest/regression/_multiple',
  body: {dependent, independents},
}).then(results => dispatch({
  type: 'RECEIVE_REGRESSION_RESULTS',
  payload: results,
}));
```

bankingIndicators.js

```
export const MONEY_IN_BANKS = 'MONEY_IN_BANKS';
export const DEPOSITS = 'DEPOSITS';
export const SHARE_CAPITAL = 'SHARE_CAPITAL';
export const NET_INCOME_AFTER_TAX_DEDUCTION =
'NET_INCOME_AFTER_TAX_DEDUCTION';
export const ALL_INDICATORS = [MONEY_IN_BANKS, DEPOSITS,
  SHARE_CAPITAL, NET_INCOME_AFTER_TAX_DEDUCTION];
```

indexReducer.js

```
import { combineReducers } from 'redux';
```

```
import regression from './regression';
const rootReducer = combineReducers({ regression });
export default rootReducer;
```

regressionReducer.js

```
const initialState = {
  regressionParameters: undefined,
  rSquared: undefined,
};

export default (state = initialState, action) => {
  switch (action.type) {
    case 'RECEIVE_REGRESSION_RESULTS': {
      return {
        regressionParameters: action.payload.parameters,
        rSquared: action.payload.rsquared,
      };
    }

    default:
      return state;
  }
};
```

RegressionContainer.jsx

```
import React, { Component } from 'react';
import { connect } from 'react-redux';
import PropTypes from 'constants/PropTypes';
import { bindActionCreators } from 'redux';
import { Checkbox } from 'material-ui';
import Button from '@material-ui/core/Button';

import * as regressionActions from '../actions/regression';
import {ALL_INDICATORS, NET_INCOME_AFTER_TAX_DEDUCTION,
SHARE_CAPITAL, DEPOSITS, MONEY_IN_BANKS} from
'../constants/bankingIndicators';

const styles = ({
  component: {
    backgroundColor: '#FFFFFF',
    borderRadius: '4px',
     boxShadow: '0px 1px 3px 0px rgba(0, 0, 0, 0.2),0px 1px 1px 0px rgba(0, 0, 0,
0.14),0px 2px 1px -1px rgba(0, 0, 0, 0.12)',
    overflow: 'hidden',
```

```
    padding: 20,
  },
  title: {
    fontWeight: 'bold',
    margin: '15px 0px',
  },
  flex: {
    display: 'flex',
  },
  description: {
    marginLeft: 10,
  },
  checkbox: {
    width: 20,
  },
  marginTop: {
    marginTop: 15,
  },
  resultDescription: {
    fontWeight: 'bold',
  },
  resultExplanation: {
    marginBottom: 8,
  }
});

const indicatorsTranslations = {
      [NET_INCOME_AFTER_TAX_DEDUCTION]: 'Net income after tax
deduction',
  [SHARE_CAPITAL]: 'Share capital',
  [DEPOSITS]: 'Deposits',
  [MONEY_IN_BANKS]: 'Money in banks',
};

class Regression extends Component {
  constructor(props) {
    super(props);
    this.state = {
      selectedDependent: undefined,
      selectedIndependents: [],
    };
  }

  render() {
    const {actions, regressionReducer} = this.props;
```

```
const independentValueName = this.state.selectedDependent
  ? indicatorsTranslations[this.state.selectedDependent]
  : '';
const rSquaredInPercent = regressionReducer.rSquared
  ? Math.round(regressionReducer.rSquared * 100)
  : undefined;
return (
  <div style={styles.component}>
    <div style={styles.title}>
      Select independent indicator
    </div>
    {ALL_INDICATORS.map(indicator => (
      <div style={styles.flex}>
        <div style={styles.checkbox}>
          <Checkbox
            checked={this.state.selectedDependent === indicator}
            disabled={this.state.selectedIndependents.includes(indicator)}
            value={indicator}
            onClick={() => this.setState({ selectedDependent: indicator })}
          />
        </div>
        <div style={styles.description}>
          {indicatorsTranslations[indicator]}
        </div>
      </div>
    ))}

    <div style={styles.title}>
      Select independent indicator/s
    </div>
    {ALL_INDICATORS.map(indicator => (
      <div style={styles.flex}>
        <div style={styles.checkbox}>
          <Checkbox
            checked={this.state.selectedIndependents.includes(indicator)}
            disabled={this.state.selectedDependent === indicator}
            value={this.state.selectedIndependents}
            onClick={() => {
              const selected = this.state.selectedIndependents;
              if (selected.includes(indicator)) {
                selected.splice(selected.indexOf(indicator), 1);
              } else {
                selected.push(indicator);
              }
              this.setState({ selectedIndependents: selected });
```

```
            }}
           />
          </div>
          <div style={styles.description}>
           {indicatorsTranslations[indicator]}
          </div>
         </div>
        ))}
        <div style={styles.marginTop}>
         <Button
          variant="contained"
          color="secondary"
          onClick={() => actions.calculate({
           dependent: this.state.selectedDependent,
           independents: this.state.selectedIndependents,
          })}
         >
          Calculate regression
         </Button>
        </div>

        {!!regressionReducer.rSquared && (
         <div>
          <div style={styles.title}>
           Regression results:
          </div>
          <div>
           {`The model explains the dependence of dependent value from the
           selected factors by ${rSquaredInPercent}%.
           It means that ${100 - rSquaredInPercent}%
           of influence are left to another factors not specified in this model.`}
          </div>
          <div style={styles.marginTop}>
              The selected independent values have the following effect on the result
value:
          </div>
          {regressionReducer.regressionParameters.map((parameter, index) => {
           const {
            bankingIndicator,
            value,
           } = parameter;
           const valueShort = value.toFixed(5);
           const parameterName = indicatorsTranslations[bankingIndicator];
           return index !== 0 && (
            <div style={styles.marginTop}>
```

```
              <div style={styles.resultDescription}>
                {parameterName}:
              </div>
              <div style={styles.resultExplanation}>
                    {`In case of ${parameterName} ${valueShort < 0 ? 'decreasing' :
'increasing'} by 1 mln UAH,
                      the value of ${independentValueName} will be increased by $
{Math.abs(valueShort)} mln UAH`}
              </div>
            </div>
          );
        })}
      </div>
    )}
  </div>
 );
 }
}

Regression.propTypes = {
 actions: PropTypes.object.isRequired,
 regressionReducer: PropTypes.object.isRequired,
};

const mapStateToProps = state => ({
 regressionReducer: state.regression.regression,
});

const mapDispatchToProps = dispatch => ({
 actions: bindActionCreators(regressionActions, dispatch)
});

export default connect(
 mapStateToProps,
 mapDispatchToProps,
)(Regression);
```

ДОДАТОК Б

Лістинг коду веб-серверу

pom.xml

```
<?xml version="1.0"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>

   <groupId>com.anna</groupId>
   <artifactId>regression</artifactId>
   <version>1.0.0</version>

   <properties>
      <java.version>1.8</java.version>
   </properties>

   <build>
      <plugins>
         <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
         </plugin>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
               <source>8</source>
               <target>8</target>
            </configuration>
         </plugin>
      </plugins>
   </build>

   <name>regression</name>
   <description>regression</description>

   <dependencies>
      <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter</artifactId>
         <version>1.5.7.RELEASE</version>
      </dependency>
```

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
   <version>1.5.7.RELEASE</version>
</dependency>

<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-context</artifactId>
   <version>4.3.11.RELEASE</version>
</dependency>

<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot</artifactId>
   <version>1.5.7.RELEASE</version>
</dependency>
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-autoconfigure</artifactId>
   <version>1.5.7.RELEASE</version>
</dependency>

<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-commons</artifactId>
   <version>1.2.2.RELEASE</version>
</dependency>

<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-web</artifactId>
   <version>4.3.11.RELEASE</version>
</dependency>

<dependency>
   <groupId>com.fasterxml.jackson.core</groupId>
   <artifactId>jackson-core</artifactId>
   <version>2.9.6</version>
</dependency>

<dependency>
   <groupId>com.fasterxml.jackson.core</groupId>
   <artifactId>jackson-databind</artifactId>
```

```
        <version>2.9.6</version>
      </dependency>

      <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-math3</artifactId>
        <version>3.6.1</version>
      </dependency>

      <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.16.20</version>
        <scope>provided</scope>
      </dependency>
    </dependencies>
</project>
```

WebConfig.java

```
package com.anna.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

/**
 * Configures web app parameters
 */
@Configuration
public class WebConfig extends WebMvcConfigurerAdapter {
  @Override
  public void addCorsMappings(CorsRegistry registry) {
    registry.addMapping("/**")
        .allowedMethods("GET", "POST", "PUT", "DELETE", "PATCH");
  }
}
```

RegressionApp.java

```
package com.anna;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
```

```
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@EnableDiscoveryClient
@ComponentScan({"com.anna"})
public class RegressionApp {
  public static void main(String[] args) throws Exception {
    SpringApplication.run(RegressionApp.class, args);
  }
}
```

RegressionController.java

```
package com.anna.rest;

import com.anna.dto.MultipleRegressionDto;
import com.anna.dto.MultipleRegressionResultDto;
import com.anna.service.RegressionService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/rest/regression")
public class RegressionController {
  @Autowired
  private RegressionService regressionService;

    @RequestMapping(value = "/_multiple", method = RequestMethod.POST, consumes = MediaType.APPLICATION_JSON_VALUE, produces = MediaType.APPLICATION_JSON_VALUE)
          public MultipleRegressionResultDto getRegression(@RequestBody MultipleRegressionDto regressionDto) {
     return regressionService.getMultipleRegression(regressionDto.getDependent(), regressionDto.getIndependents());
  }
}
```

RegressionService.java

```
package com.anna.service;

import com.anna.dict.BankingIndicators;
```

```
import com.anna.dto.MultipleRegressionResultDto;

import java.util.List;

public interface RegressionService {
        MultipleRegressionResultDto getMultipleRegression(BankingIndicators
dependent, List<BankingIndicators> independents);
}
```

RegressionServiceJdbc.java

```
package com.anna.service;

import com.anna.dao.RegressionDao;
import com.anna.dict.BankingIndicators;
import com.anna.dto.MultipleRegressionResultDto;
import com.anna.dto.RegressionParametersValuesDto;
import org.apache.commons.math3.stat.regression.OLSMultipleLinearRegression;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;

@Service
public class RegressionServiceJdbc implements RegressionService {

  @Autowired
  private RegressionDao regressionDao;

  @Override
    public MultipleRegressionResultDto getMultipleRegression(BankingIndicators
dependent, List<BankingIndicators> independents) {
    List<Double> dependentValues = getListByIndicator(dependent);
                        List<List<Double>> independentValues = new
ArrayList<>(independents.size());
                                    independents.forEach(indicator ->
independentValues.add(getListByIndicator(indicator)));

    List<Double> finalValues = new ArrayList<>();

    for (int i = 0; i < dependentValues.size(); i++) {
      finalValues.add(dependentValues.get(i));
      for (List<Double> independentValue : independentValues) {
        finalValues.add(independentValue.get(i));
```

```
    }
   }

            OLSMultipleLinearRegression multipleLinearRegression = new
OLSMultipleLinearRegression();
  multipleLinearRegression.newSampleData(finalValues.stream().mapToDouble(d
-> d).toArray(), dependentValues.size(), independents.size());

   double rSquared = multipleLinearRegression.calculateRSquared();
                                    double[] regressionParameters =
multipleLinearRegression.estimateRegressionParameters();

         List<RegressionParametersValuesDto> regressionParamsValues = new
ArrayList<>(independents.size() + 1);
     regressionParamsValues.add(new RegressionParametersValuesDto(dependent,
regressionParameters[0]));
   for (int i = 1; i < regressionParameters.length; i++) {
     double value = regressionParameters[i];
     BankingIndicators indicator = independents.get(i - 1);
       regressionParamsValues.add(new RegressionParametersValuesDto(indicator,
value));
   }
return MultipleRegressionResultDto.builder()
     .parameters(regressionParamsValues)
     .rSquared(rSquared)
     .build();
  }

  private List<Double> getListByIndicator(BankingIndicators indicator) {
   switch (indicator) {
     case DEPOSITS: return regressionDao.getDeposites();
     case SHARE_CAPITAL: return regressionDao.getShareCapital();
     case MONEY_IN_BANKS: return regressionDao.getMoneyInBanks();
                    case NET_INCOME_AFTER_TAX_DEDUCTION: return
regressionDao.getNetIncomeAfterTaxDeduction();
            default: throw new RuntimeException("Incorrect indicator: " +
indicator.name());
   }
  }
}
```

RegressionDao.java

```
package com.anna.dao;
```

```
import java.util.List;

public interface RegressionDao {
  List<Double> getMoneyInBanks();
  List<Double> getDeposites();
  List<Double> getShareCapital();
  List<Double> getNetIncomeAfterTaxDeduction();
}
```

RegressionJdbcDao.java

```
package com.anna.dao;

import org.springframework.stereotype.Component;

import java.util.Arrays;
import java.util.List;

@Component
public class RegressionJdbcDao implements RegressionDao {
  public List<Double> getMoneyInBanks() {
    return Arrays.asList(
      551081281.15d, 500860529.59d, 415443523.08d, 411282079.10d,
      381724868.28d, 377566292.38d, 221131028.11d, 172764481.07d,
      215217184.25d, 228755796.19d, 315234821.82d, 297086846.11d,
      223020138.24d, 193044277.32d, 244301720.58d, 216385531.48d,
      187008114.47d, 141509629.45d, 177420102.26d, 189564075.86d,
      230806824.93d, 228353978.22d, 182576806.70d, 159371480.78d
    );
  }

  public List<Double> getDeposites() {
    return Arrays.asList(
      446218040.25d, 455611168.48d, 397529127.44d, 373019155.45d,
      375020937.28d, 364768147.13d, 258811086.73d, 193742852.84d,
      403015743.77d, 172600504.01d, 146110295.67d, 175507614.64d,
      189759050.78d, 172959227.53d, 219144342.97d, 224253777.06d,
      253354325.80d, 234861036.76d, 197363352.71d, 254468651.77d,
      209358375.68d, 177551612.17d, 200682728.33d, 149670906.72d
    );
  }

  public List<Double> getShareCapital() {
    return Arrays.asList(
      475429794.04d, 504832668.48d, 511951241.91d, 510821763.32d,
```

```
      511921836.90d, 511810991.88d, 507748159.66d, 518177900.86d,
      507936111.36d, 525031820.09d, 513559637.91d, 506195901.62d,
      519113892.48d, 507473774.63d, 509751030.49d, 507862333.48d,
      510441764.21d, 503958279.70d, 499706112.94d, 510909698.38d,
      509962205.09d, 512407895.31d, 510007520.31d, 507719271.55d
    );
  }

  public List<Double> getNetIncomeAfterTaxDeduction() {
    return Arrays.asList(
      -22460985.10d, 7780634.35d, 12531908.81d, 13598867.02d,
      15166300.18d, 14584161.02d, 14856236.96d, 16816664.16d,
      10251664.31d, 23593663.50d, 12212623.37d, 8157704.89d,
      12925708.21d, 11302049.28d, 9715430.18d, 14652556.31d,
      20798752.06d, 10466409.91d, 8571597.36d, 14435582.53d,
      11497880.94d, 14153052.63d, 14913817.00d, 9561518.03d
    );
  }
}
```

MultipleRegressionDto.java

```
package com.anna.dto;

import com.anna.dict.BankingIndicators;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;

@Data
@NoArgsConstructor
public class MultipleRegressionDto {
  BankingIndicators dependent;
  List<BankingIndicators> independents;
}
```

BankingIndicators.java

```
package com.anna.dict;

public enum BankingIndicators {
  MONEY_IN_BANKS,
  DEPOSITS,
  SHARE_CAPITAL,
  NET_INCOME_AFTER_TAX_DEDUCTION,
```

```
}
```

RegressionParametersValuesDto.java

```
package com.anna.dto;

import com.anna.dict.BankingIndicators;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class RegressionParametersValuesDto {
  private BankingIndicators bankingIndicator;
  private double value;
}
```

MultipleRegressionResultDto.java

```
package com.anna.dto;

import lombok.Builder;
import lombok.Data;

import java.util.List;

@Data
@Builder(toBuilder = true)
public class MultipleRegressionResultDto {
  private double rSquared;
  private List<RegressionParametersValuesDto> parameters;
}
```

ДОДАТОК В

Слайди презентації

ДОСЛІДЖЕННЯ МЕТОДІВ АНАЛІЗУ ДАНИХ ЗА ДОПОМОГОЮ ЧАСОВИХ РЯДІВ ДЛЯ СТВОРЕННЯ БАНКІВСЬКОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Виконала: студентка 2 курсу, групи ІПЗ-мзд-17-1
спеціальності 121- Інженерія програмного забезпечення
ОНП Інженерія програмного забезпечення
Чепіга А.О.

АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

Аналіз діяльності банків:

- відправний пункт ефективного управління як окремим комерційним банком, так і банківською системою в цілому,
- вихідна база для прийняття управлінських рішень на всіх рівнях.

Інструментарій:

- аналітичний модуль у складі автоматизованих банківських систем,
- зробити процес та результат аналізу показників діяльності банку автоматизованим, простим та наглядним.

Об'єкт дослідження - методи аналізу даних за допомогою часових рядів в сфері банківської діяльності.

Предмет дослідження - специфіка використання методів аналізу даних за допомогою часових рядів в сфері банківської діяльності.

Мета дослідження - проаналізувати специфіку та способи використання методів аналізу даних за допомогою часових рядів у складі банківської інформаційної системи.

ЧАСОВИЙ РЯД

Послідовність, взята на рівновіддалених точках в часі, які йдуть одна за одною.

Використовуються в:

- статистиці,
- обробці сигналів,
- розпізнаванні образів,
- фінансовій математиці,
- в будь-якій області прикладної науки та інженерії, яка включає часові вимірювання.

АНАЛІЗ ЧАСОВИХ РЯДІВ
(англ. time series analysis)

Включає методи аналізу даних часових рядів з метою витягування значимих статистик та інших характеристик даних.

Метод аналізу часового ряду визначається:

- цілями аналізу,
- ймовірнісною природою формування його значень.

МЕТОДИ АНАЛІЗУ ЧАСОВИХ РЯДІВ

- Спектральний
- Кореляційний
- Моделі авторегресії і ковзного середнього
- Багатоканальні моделі авторегресії і ковзного середнього
- Сезонна модель Бокса-Дженкінса
- Прогноз експоненціально зваженим ковзаючим середнім

АВТОМАТИЗОВАНА БАНКІВСЬКА СИСТЕМА (АБС)

Система, яка функціонує на основі ЕОМ та інших технічних засобів, що забезпечують процеси:

- збору,
- реєстрації,
- передачі,
- обробки,
- збереження,
- актуалізації даних

для розв'язання завдань управління банківською діяльністю.

ПІДСИСТЕМА "АНАЛІЗ ДІЯЛЬНОСТІ БАНКУ"

Акумулює у своєму складі аналітичні задачі, які належать до класу OLAP.

OLAP (АНГЛ. ONLINE ANALYTICAL PROCESSING, АНАЛІТИЧНА ОБРОБКА У РЕАЛЬНОМУ ЧАСІ)

Інтерактивна система, що дозволяє переглядати різні підсумки по багатовимірних даних.

УКРАЇНСЬКІ АБС

Більшість українських АБС в аналітичному модулі мають функціонал, більшість операцій якого пов'язані із формуванням звітів, а не аналізом статистичної інформації та не з побудовою прогнозів.

Мають аналітичний модуль

- АБС «Б2»
- АБС «UNITY-BARS»
- АБС «SRBank»

Не мають аналітичний модуль

- АБС«SCROOGE»
- АБС«ProFIX»
- АБС«ГРАНТ»

Аналіз показників діяльності банку має велику цінність, але в той же час проведення подібного аналізу вимагає великої кількості дій та етапів, що не є зручним для кінцевого споживача, тобто банківського співробітника, чиєю діяльністю є проведення подібного аналізу.

Регресійна статистика					
Множинний R	0,988752269				
R-квадрат	0,881255822				
Нормований R-квадрат	0,863444195				
Стандартна похибка	3020164,661				
Спосте-реження	24				
Дисперсійний аналіз					
	df	*SS*	*MS*	*F*	*Значимість F*
Регресія	3	1,35388E+15	4,51294E+14	49,47643684	1,94773E-09
Залишок	20	1,82428E+14	9,12139E+12		
Усього	23	1,53631E+15			

Y = 0,797 x3 - 0,0256x1 - 391978977,9

Спрощення процедури: модулі статистичного аналізу даних в існуючих АБС:

- обробка сукупності даних,
- простий та зрозумілий результат,
- не вимагати від людини мануального проходження всіх необхідних етапів.

Розроблюваний додаток - **функціонал**:

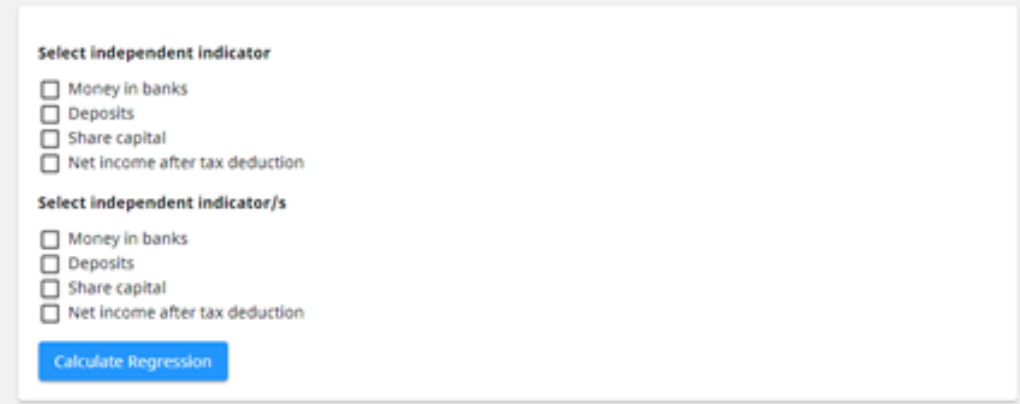
- відображати список банківських показників, які можна аналізувати;
- можливість вибирати з цих показників один залежний та декілька незалежних;
- розрахунки та зрозуміла і вже інтерпретована інформація.

Обгрунтування: відсутність необхідності самостійно інтерпретувати набір отриманих чисел (результатів аналізу), а замість того можливість дізнатися результати одразу з економічної точки зору.

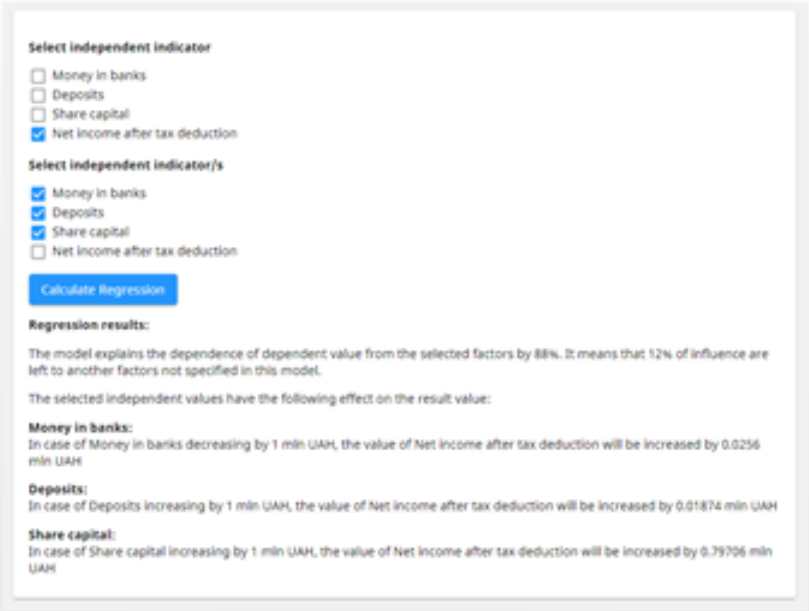
ДЕМОНСТРАЦІЯ ПРОДУКТУ

Головна сторінка із можливістю обирати

залежні та незалежні показники



РЕЗУЛЬТАТИ ОБРОБКИ



Інтерфейс користувача

- React JS;
- Node.js;
- Redux;
- Material UI.

Серверна частина додатку

- Java;
- Spring;
- Spring Boot;
- Apache Commons Math;
- Lombok.

ВИСНОВКИ

Для того, аби отримати та інтерпретувати результати аналізу, потрібен фахівець, здатний

- розуміти велику кількість математичних та статистичних термінів,
- вміти обробляти результати на проміжному рівні,
- інтерпретувати дані виходячи з проміжних результатів.

Розроблюваний продукт має ознаки новизни з точки зору бізнесу.

Є сенс в його подальшому розвитку та удосконаленні із урахуванням специфіки тих автоматичних банківських систем, які б його використовували.

ДОДАТОК Г

Апробація результатів роботи

Тези «Дослідження методів аналізу даних за допомогою часових рядів для створення банківської інформаційної системи»

MONOGRAFIA
POKONFERENCYJNA

SCIENCE,
RESEARCH, DEVELOPMENT #17

TECHNICS AND TECHNOLOGY.

Belgrade (Serbia)

30.05.2019- 31.05.2019

U.D.C. 330+339.138+658+657+336.71+339+082
B.B.C. 94
Z 40

Zbiór artykułów naukowych recenzowanych.
(1) Z 40 Zbiór artykułów naukowych z Konferencji Miedzynarodowej Naukowo-Praktycznej (on-line) zorganizowanej dla pracowników naukowych uczelni, jednostek naukowo-badawczych oraz badawczych z państw obszaru byłego Związku Radzieckiego oraz byłej Jugosławii.

(*30.05.2019*) - Warszawa, 2019. - 84 str.
ISBN: 978-83-66401-02-0

Wydawca: Sp. z o.o. «Diamond trading tour»
Adres wydawcy i redakcji: 00-728 Warszawa, ul. S. Kierbedzia, 4 lok.103
e-mail: info@conferenc.pl

W artykułach naukowych zachowano oryginalną pisownię.

Wszystkie artykuły naukowe są recenzowane przez dwóch członków Komitetu Naukowego.

W przypadku cytowań obowiązkowe jest odniesienie się do monografii.

Nakład: 80 egz.

ISBN: 978-83-66401-02-0

ДОСЛІДЖЕННЯ МЕТОДІВ АНАЛІЗУ ДАНИХ ЗА ДОПОМОГОЮ ЧАСОВИХ РЯДІВ ДЛЯ СТВОРЕННЯ БАНКІВСЬКОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Чепіга А.О.
Харківський національний університет радіоелектроніки

Ключові слова: автоматизовані банківські системи, методи аналізу даних, показники банківської діяльності, часові ряди, Apache Common Math, Java, React.js, web-сайт.

Keywords: Apache Common Math, automated banking systems, indicators of banking activities, Java, methods of data analysis, React.JS, time series, web-site.

Кожен день банківські установи, зважаючи на специфіку їх діяльності, наражаються на велику кількість фінансових ризиків. Аналіз діяльності банку є основою ефективного управління ним, вихідною базою прийняття управлінських рішень. За допомогою такого аналізу розробляють стратегію і тактику розвитку банку, обґрунтовують плани й управлінські рішення, здійснюють контроль за їх виконанням, виявляють резерви підвищення ефективності проведення активних і пасивних операцій, оцінюють результати діяльності банку, його окремих підрозділів.

Сучасні банки – це складні IT-системи. Автоматизована банківська система (ABS або core banking) є загальноприйнятим терміном, який зазвичай означає набір комп'ютерних програм або програмного забезпечення, призначеного для інтеграції автоматизації банківської діяльності, є повноцінним інструментом для банківського бізнесу і дозволяє автоматизувати широкий спектр бізнес-процесів та фінансових інструментів банку.

Актуальність дослідження обумовлена тим, що аналіз діяльності банків є відправним пунктом ефективного управління як окремим комерційним банком, так і банківською системою в цілому, а також вихідною базою для прийняття управлінських рішень на всіх рівнях; але в той же час для проведення подібного аналізу необхідний відповідний інструментарій, тобто аналітичний модуль у складі автоматизованих банківських систем, за допомогою якого процес та результат аналізу показників діяльності банку був би автоматизованим, простим та наглядним.

Аналіз показників діяльності банку має велику цінність, але в той же час проведення подібного аналізу вимагає великої кількості дій та етапів, що не є зручним для кінцевого споживача, тобто банківського співробітника, чиєю діяльністю є проведення подібного аналізу [1, 8]. Наше дослідження показало, що для того, аби отримати та інтерпретувати результати аналізу, потрібен фахівець, здатний розуміти велику кількість математичних та ста-

тистичних термінів, вміти обробляти результати на проміжному рівні, а також інтерпретувати дані виходячи з проміжних результатів. Цю процедуру можна було би спростити, використовуючи модулі статистичного аналізу даних в складі автоматизованих банківських систем, які обробляли би сукупність даних та виводили простий та зрозумілий результат, не вимагаючи від людини мануального проходження всіх необхідних етапів.

В той же час, аналіз найбільш відомих існуючих українських автоматизованих банківських систем («Б2», «UNITY-BARS», «SCROOGE», «SR-Bank», «ProFIX», «ГРАНТ») показав, що аналітичними модулями володіють АБС «Б2», АБС «UNITY-BARS», АБС «SRBank». Такі АБС, як «SCROOGE», «ProFIX» та «ГРАНТ», не включать модулів щодо аналітики показників діяльності банку. З іншого боку, окрім власне аналітики, не менш важливим є і прогноз майбутніх показників, і згідно нашого дослідження, подібний функціонал майже відсутній у всіх АБС, що підтверджує необхідність його реалізації [2, 3, 4, 5, 6, 7].

Крім того, наочний приклад використання методів аналізу даних за допомогою часових рядів для оцінки ефективності діяльності окремого обраного банку обґрунтував необхідність кількісного виміру ступеню впливу відповідних факторів та визначення важливості застосування багатофакторного кореляційно-регресійного аналізу в якості найбільш ефективного методу економіко-математичного моделювання, що відображає характер взаємозв'язку між ознаками досліджуваного явища; та заклав структуру та ідею майбутнього програмного продукту.

У практичній частині роботи було проілюстровано демо майбутнього програмного продукту у скороченій версій, для демонстрації основного функціоналу. Демо-версія додатку була розроблена за допомогою таких технологій: бібліотеки React JS, платформи Node.js, бібліотеки Redux, бібліотеки Material UI – для створення веб-клієнту; серверна частина додатку написана на мові програмування Java із використанням фреймворку Spring, технології Spring Boot, бібліотеки Lombok, бібліотеки Apache Commons Math для проведення розрахунків. Враховуючи проведені дослідження, ми вважаємо, що подібний продукт був би корисним на ринку та користовувався би попитом, отож є сенс в його подальшому розвитку та удосконаленні із урахуванням специфіки тих автоматичних банківських систем, які б його використовували.

1. А.О. Криштановська. Методи аналізу часових рядів / / Моніторинг громадської думки: економічні і соціальні зміни. 2000. № 2 (46). С. 44-51.
2. АБС «ProFIX» [Електронний ресурс]/ Режим доступу: URL: http://profix.com.ua/ua/
3. АБС «SCROOGE» [Електронний ресурс]/ Режим доступу: URL: http://lime-systems.com/products/abs-scrooge/
4. АБС «SRBank» [Електронний ресурс]/ Режим доступу: URL: http://

www.soft-review.com.ua/about/

5. АБС «UNITY-BARS» [Електронний ресурс]/ Режим доступу: URL: https://www.unity-bars.com/absbars/
6. АБС «Б2» [Електронний ресурс]/ Режим доступу: URL: https://csltd.com.ua/products/core_banking
7. АБС «БІС ГРАНТ» [Електронний ресурс]/ Режим доступу: URL: http://www.banksoft.com.ua/uk/#
8. Бокс Дж., Дженкинс Г. Анализ временных рядов, прогноз и управление: Пер. с англ. // Под ред. В. Ф. Писаренко. – М.: Мир, 1974.