

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
 «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

**Лабораторная работа № 5**

**Тема: «Машинное обучение»**

**Дисциплина: «Формальные языки»**

Выполнил:

студент группы ВКБ-21

Котелевец К.А.

Проверил:

доцент Пиневич Е.В.

Ростов-на-Дону

2024

Цель:

- закрепить понятия «Элементы машинного обучения», «Алгоритм “Наивный Байес”»

- Разработать программное средство, реализующий алгоритм Наивного Байеса

Постановка задачи к лабораторной работе №5:

**Дано:**

Система, которая обучалась на следующих письмах, заранее известных, где спам – не спам.

Спам сообщения:

1. Поздравляем, ты выиграл приз
2. Купи товар прямо сейчас и получи скидку!
3. Привет, это сотрудник банка, назовите пин код карты

Не спам сообщения:

1. Привет, как дела?
2. Давай прогуляемся этим вечером
3. Сходи сейчас в магазин и купи там две булки хлеба
4. Я сегодня выиграл приз и получил скидку двадцать пять процентов

**Задача:**

1. Определить к какой категории относиться следующее сообщение:  
   *Сейчас давай сходим в магазин, я схожу куплю три булки хлеба, а ты купи игральные карты. Будем показывать фокусы и розыгрыш на большой приз.*
2. Разработать программное средство, реализующее Наивный байесовский классификатор

**Ход работы:**

P(H1) = - Спам

P(H2) = – Не спам

Коэффициентом сглаживания пример значение 1.

Построим таблицу, состоящую из слов, без предлогов и заполним (Таблица 1).

Таблица 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Слово | Число вхождений в Спам | Число вхождений в Не спам | P(спам) (Ai) | P(не спам) (Ai) |
| **сейчас** | 1 | 1 |  |  |
| получил | 0 | 1 |  |  |
| поздравляем | 1 | 0 |  |  |
| назовите | 1 | 0 |  |  |
| пять | 0 | 1 |  |  |
| прямо | 1 | 0 |  |  |
| скидку | 1 | 1 |  |  |
| **куплю** | 0 | 0 |  |  |
| **игральные** | 0 | 0 |  |  |
| **ты** | 1 | 0 |  |  |
| привет | 1 | 1 |  |  |
| **давай** | 0 | 1 |  |  |
| **я** | 0 | 1 |  |  |
| сотрудник | 1 | 0 |  |  |
| пин | 1 | 0 |  |  |
| **карты** | 1 | 0 |  |  |
| как | 0 | 1 |  |  |
| **схожу** | 0 | 0 |  |  |
| **булки** | 0 | 1 |  |  |
| **хлеба** | 0 | 1 |  |  |
| процентов | 0 | 1 |  |  |
| выиграл | 1 | 1 |  |  |
| это | 1 | 0 |  |  |
| **магазин** | 0 | 1 |  |  |
| этим | 0 | 1 |  |  |
| **фокусы** | 0 | 0 |  |  |
| **приз** | 1 | 1 |  |  |
| **купи** | 1 | 1 |  |  |
| там | 0 | 1 |  |  |
| **розыгрыш** | 0 | 0 |  |  |
| код | 1 | 0 |  |  |
| две | 0 | 1 |  |  |
| сегодня | 0 | 1 |  |  |
| двадцать | 0 | 1 |  |  |
| **три** | 0 | 0 |  |  |
| **большой** | 0 | 0 |  |  |
| **сходим** | 0 | 0 |  |  |
| сходи | 0 | 1 |  |  |
| дела | 0 | 1 |  |  |
| получи | 1 | 0 |  |  |
| **а** | 0 | 0 |  |  |
| прогуляемся | 0 | 1 |  |  |
| вечером | 0 | 1 |  |  |
| товар | 1 | 0 |  |  |
| **и** | 1 | 2 |  |  |
| **будем** | 0 | 0 |  |  |
| **показывать** | 0 | 0 |  |  |
| банка | 1 | 0 |  |  |

=> Спам < Не спам

Наше сообщение относится к классификации “НЕ СПАМ”

Листинг Программы

Word.class

|  |
| --- |
| package com.kiku.formallanguagemachineteaching;  public class Word {   public Word(String word, Integer spam, Integer noSpam, Double pSpam, Double pNoSpam) {  this.word = word;  this.spam = spam;  this.noSpam = noSpam;  this.pSpam = pSpam;  this.pNoSpam = pNoSpam;  }     public String getWord() {  return word;  }   public Integer getSpam() {  return spam;  }   public Integer getNoSpam() {  return noSpam;  }   public Double getpSpam() {  return pSpam;  }   public Double getpNoSpam() {  return pNoSpam;  }   public Double getPSpam() {  return pSpam;  }   public Double getPNoSpam() {  return pNoSpam;  }   String word;  Integer spam;  Integer noSpam;  Double pSpam;  Double pNoSpam; } |

Message.class

|  |
| --- |
| package com.kiku.formallanguagemachineteaching;  public class Message {  String message;  String status;    public Message(String message, String status) {  this.message = message;  this.status = status;  }   public String getMessage() {  return message;  }   public String getStatus() {  return status;  }  } |

MainApp:

|  |
| --- |
| package com.kiku.formallanguagemachineteaching;  import javafx.application.Application; import javafx.fxml.FXMLLoader; import javafx.scene.Scene; import javafx.stage.Stage;  import java.io.IOException;  public class MainApp extends Application {  @Override  public void start(Stage stage) throws IOException {  FXMLLoader fxmlLoader = new FXMLLoader(MainApp.class.getResource("mainApp.fxml"));  Scene scene = new Scene(fxmlLoader.load());  stage.setTitle("Наивный Байес - Машинное обучение");  stage.setScene(scene);  stage.show();  }   public static void main(String[] args) {  *launch*();  } } |

DbConnect

|  |
| --- |
| package com.kiku.formallanguagemachineteaching;  import java.sql.Connection; import java.sql.DriverManager; import java.sql.SQLException; import java.util.logging.Level; import java.util.logging.Logger;  public class DbConnect {  private static Connection *connection*;    *// Создание соединения к базе данных* public static Connection getConnect (){  try {  String password = "root";  String user = "root";  String DB\_NAME = "baiesfl";  int PORT = 3306;  String HOST = "127.0.0.1";  *connection* = DriverManager.*getConnection*(String.*format*("jdbc:mysql://%s:%d/%s", HOST, PORT, DB\_NAME), user, password);  } catch (SQLException ex) {  Logger.*getLogger*(DbConnect.class.getName()).log(Level.*SEVERE*, null, ex);  }   return *connection*;  } } |

BaiesController:

|  |
| --- |
| package com.kiku.formallanguagemachineteaching;  import javafx.collections.FXCollections; import javafx.collections.ObservableList; import javafx.scene.control.\*; import javafx.scene.control.cell.PropertyValueFactory; import javafx.scene.paint.Color;  import java.sql.Connection; import java.sql.PreparedStatement; import java.sql.ResultSet; import java.sql.SQLException; import java.util.\*;   @SuppressWarnings("ALL") public class BaiesController {   public Button removeButton;  public TextArea otherMessageField;  public CheckBox checkBoxSpam;  public Button addButton;  public TextField pSpamField;  public TextField pNoSpamField;  public Label stavniLabel;  public TextField clearWordField;  Double countSpamP;  Double countNoSpamP;   public TableView<Word> wordsTable;  public TableColumn<Word, String> wordColumn;  public TableColumn<Word, Integer> spamColumn;  public TableColumn<Word, Integer> noSpamColumn;  public TableColumn<Word, Double> pSpamColumn;  public TableColumn<Word, Double> pNoSpamColumn;  public TextArea messageField;  public TextField statusFiled;  public Button calculateButton;  public TableView<Message> messageTable;  public TableColumn<Message, String> messageColumn;  public TableColumn<Message, String> statusColumn;  public TextField coffField;    Connection connection = DbConnect.*getConnect*();  ObservableList<Message> MessageList = FXCollections.*observableArrayList*();   ObservableList<Word> WordList = FXCollections.*observableArrayList*();   public void initialize() throws SQLException {   messageField.setText(" ");  coffField.setText("1.0");  loadDateToMessageTable();  messageField.clear();   messageTable.setOnMouseClicked(event -> {  if (event.getClickCount() == 1) {  Message selectedMessage = messageTable.getSelectionModel().getSelectedItem();   if (selectedMessage != null) {  String messageContent = selectedMessage.getMessage();  otherMessageField.setText(messageContent);  checkBoxSpam.setSelected(selectedMessage.getStatus().equals("SPAM"));  }  }  });    */\*  \* Обрабатывает таблицу, закрашивая выделенные ячейки со словами исследуюемого сообщения зеленым цветом  \* \*/* wordColumn.setCellFactory(column -> {  return new TableCell<Word, String>() {  @Override  protected void updateItem(String item, boolean empty) {  super.updateItem(item, empty);   if (item == null || empty) {  setText(null);  setStyle("");  } else {  setText(item);   String messageFieldText = messageField.getText();  if (messageFieldText != null && !messageFieldText.isEmpty() && item != null && !item.isEmpty()) {  boolean containsWholeWord = false;  String[] words = messageFieldText.split("\\s+");  for (String word : words) {  if (word.equalsIgnoreCase(item)) {  containsWholeWord = true;  break;  }  }   if (containsWholeWord) {  setStyle("-fx-background-color: lightgreen;");  setTextFill(Color.*BLACK*);  } else {  setStyle("");  setTextFill(Color.*BLACK*);  }  } else {  setStyle("");  setTextFill(Color.*BLACK*);  }  }  }  };  });    }   */\* Метод для подсчета количества слов в сообщении  \* @param word - слово, которое нужно подсчитать  \* @param Messages - список сообщений, в которых нужно искать слово  \* @return количество вхождений слова в сообщения  \*/* public Integer countWordInMessages(String word, List<String> Messages) {  int count = 0;  String regex = "(^|\\s+)" + word + "(\\s+|$)";   for (String message : Messages) {  if (message.matches(".\*" + regex + ".\*")) {  count++;  }  }  return count;  }    */\*\*  \* Обновляет таблицу сообщений, извлекая сообщения из базы данных и обновляя пользовательский интерфейс.  \*  \* @throws SQLException если происходит ошибка доступа к базе данных  \*/* public void refreshMessageTable() throws SQLException {   MessageList.clear();    PreparedStatement preparedStatement = connection.prepareStatement("SELECT \* FROM baiesfl.message");  ResultSet resultSet = preparedStatement.executeQuery();   while (resultSet.next()) {  MessageList.add(new Message(resultSet.getString("message").toLowerCase(), resultSet.getString("status")));   messageTable.setItems(MessageList);  }    }    */\* Загрузка сообщений в таблицу  \* @throws SQLException если происходит ошибка доступа к базе данных  \*/* public void loadDateToMessageTable() throws SQLException {   refreshMessageTable();    messageColumn.setCellValueFactory(new PropertyValueFactory<>("message"));  statusColumn.setCellValueFactory(new PropertyValueFactory<>("status"));  fillWordColumn();  }    */\*\*  \* Метод для извлечения всех сообщений с определенноым статусом из MessageList.  \*  \* @return Возвращает список сообщений после обработки.  \*/* public List<String> getMessages(String status) {  List<String> spamMessages = new ArrayList<>();   for (Message message : MessageList) {  if (message.getStatus().equals(status)) {  spamMessages.add(*removePunctuation*(removePrepositions(message.getMessage())));  }   }  return spamMessages;  }     */\* Удаление знаков препинания из сообщений  \* @param message - сообщение, из которого нужно удалить знаки препинания  \* @return сообщение без знаков препинания  \*/* public String removePrepositions(String message) {  List<String> prepositions = Arrays.*asList*(clearWordField.getText().split("\\s+"));   String[] words = message.split("\\s+");  StringBuilder result = new StringBuilder();   for (String word : words) {  if (!prepositions.contains(word.toLowerCase())) {  result.append(word).append(" ");  }  }   return result.toString().trim();  }   */\* Этот фрагмент кода представляет собой метод на Java,  \* который заполняет столбец слов в таблице. Он обрабатывает  \* сообщения, отмеченные как "СПАМ" и "НЕ СПАМ", извлекает слова,  \* вычисляет вероятности и создает объекты Word для каждого  \* уникального слова для отображения в таблице.  \*/* public void fillWordColumn() {  Set<String> uniqueWords = new HashSet<>();  Set<String> spamWords = new HashSet<>();  Set<String> noSpamWords = new HashSet<>();  List<String> spamMessages = getMessages("SPAM");  List<String> noSpamMessages = getMessages("NO SPAM");    String[] wordsCorrectMessage = removePrepositions(*removePunctuation*(messageField.getText().toLowerCase())).split("\\s+");    *// Проход по каждой строке в messageColumn* for (String message : spamMessages) {  spamWords.addAll(Arrays.*asList*(removePrepositions(message).split("\\s+")));  }   for (String message : noSpamMessages) {  noSpamWords.addAll(Arrays.*asList*(removePrepositions(message).split("\\s+")));  }   uniqueWords.addAll(spamWords);  uniqueWords.addAll(noSpamWords);    int countM = uniqueWords.size();   int countSpam = spamWords.size();   int countNoSpam = noSpamWords.size();   Double coff = Double.*parseDouble*(coffField.getText());   countSpamP = getP(MessageList.size(), spamMessages.size());   countNoSpamP = getP(MessageList.size(), noSpamMessages.size());    uniqueWords.addAll(Arrays.*asList*(wordsCorrectMessage));    *// Создание объектов Word для каждого уникального слова и добавление их в список* for (String word : uniqueWords) {  Integer countSpamOfWord = countWordInMessages(word, spamMessages);  Integer countNoSpamOfWord = countWordInMessages(word, noSpamMessages);   Double pSpam = (coff + countSpamOfWord) / (coff \* countM + countSpam);   Double pNoSpam = (coff + countNoSpamOfWord) / (coff \* countM + countNoSpam);   if (Arrays.*asList*(wordsCorrectMessage).contains(word)) {  WordList.add(new Word(*removePunctuation*(word), countSpamOfWord, countNoSpamOfWord, pSpam, pNoSpam));  } else {  WordList.add(new Word(*removePunctuation*(word), countSpamOfWord, countNoSpamOfWord, 1.0, 1.0));   }  }   *// Установка списка уникальных слов в таблицу* wordsTable.setItems(WordList);    wordColumn.setCellValueFactory(new PropertyValueFactory<>("word"));  spamColumn.setCellValueFactory(new PropertyValueFactory<>("spam"));  noSpamColumn.setCellValueFactory(new PropertyValueFactory<>("noSpam"));  pSpamColumn.setCellValueFactory(new PropertyValueFactory<>("pSpam"));  pNoSpamColumn.setCellValueFactory(new PropertyValueFactory<>("pNoSpam"));    }   */\*Этот Java код определяет метод calculate, который выполняет  различные операции, такие как обновление таблицы сообщений,  очистку списка слов, заполнение столбца слов, установку текста  в определенных полях на основе вычислений, определение статуса  сообщения, установку стилей и обновление полей сообщений.  Метод также обновляет поле статуса, метку и флажок на основе результатов вычислений спама.  \*/* public void calculate() throws SQLException {    refreshMessageTable();  WordList.clear();  fillWordColumn();    pSpamField.setText(String.*valueOf*(getArg(pSpamColumn, countSpamP)));  pNoSpamField.setText(String.*valueOf*(getArg(pNoSpamColumn, countNoSpamP)));   */\* Определяем статус сообщения и выставляем соответствующий стиль \*/* if (getArg(pSpamColumn, countSpamP) > getArg(pNoSpamColumn, countNoSpamP)) {  statusFiled.setText("SPAM");  statusFiled.setStyle("-fx-background-color: red;");  checkBoxSpam.setSelected(true);   stavniLabel.setText("⋁");  } else {  statusFiled.setText("NO SPAM");  statusFiled.setStyle("-fx-background-color: green;");  checkBoxSpam.setSelected(false);  stavniLabel.setText("⋀");  }    otherMessageField.setText(messageField.getText());  addMessage();    }    */\* Удаление знаков препинания из слова или сообщения \*/* public static String removePunctuation(String word) {   List<String> prepositions = Arrays.*asList*("в", "на", "из", "от", "до", "по", "за", "под", "над", "к", "с", "у", "о", "об", "перед", "через", "между", "подо");    String cleanWord = word.replaceAll("\\p{Punct}", "").toLowerCase();    for (String preposition : prepositions) {  cleanWord = cleanWord.replaceAll("\\b" + preposition + "\\b", "");  }   return cleanWord.trim();  }   */\* Получения вероятности \*/* public Double getP(int countM, int count) {   return count \* 1.0 / countM;  }    */\* Вычисление аргумента \*/* public Double getArg(TableColumn<Word, Double> column, Double p) {   double result = 1.0;    for (Word word : wordsTable.getItems()) {  Double pSpamValue = column.getCellData(word);    result \*= pSpamValue;  }   result \*= p;    return result;  }    */\* Удаление сообщения из БД SQL \*/* public void removeMessage() {   try {  String query = "DELETE FROM baiesfl.message WHERE message = ?";  PreparedStatement preparedStatement = connection.prepareStatement(query);  preparedStatement.setString(1, otherMessageField.getText()); *// Assuming id is the primary key of the message table* preparedStatement.executeUpdate();   *// Refresh the table after deletion* refreshMessageTable();  } catch (SQLException e) {  System.*out*.println("Error deleting message: " + e.getMessage()); *// Handle any SQLprintln(); // Handle any SQL exceptions* }   otherMessageField.clear();  checkBoxSpam.setSelected(false);   }   */\* Добавление сообщения в БД SQL \*/* public void addMessage() throws SQLException {   if (!checkMessageExists(otherMessageField.getText())) {   String query = "INSERT INTO baiesfl.message (message, status) VALUES (?, ?)";  PreparedStatement preparedStatement = connection.prepareStatement(query);  preparedStatement.setString(1, otherMessageField.getText());  if (checkBoxSpam.isSelected()) {  preparedStatement.setString(2, "SPAM");  } else {  preparedStatement.setString(2, "NO SPAM");  }   preparedStatement.executeUpdate();   refreshMessageTable();  otherMessageField.clear();  checkBoxSpam.setSelected(false);  } else {  otherMessageField.clear();  checkBoxSpam.setSelected(false);  statusFiled.setText("ОШИБКА");  statusFiled.setStyle("-fx-background-color: yellow;");  }  }   */\* Проверка наличия сообщения в БД SQL \*/* public boolean checkMessageExists(String messageContent) throws SQLException {  String query = "SELECT COUNT(\*) AS count FROM baiesfl.message WHERE message = ?";  PreparedStatement preparedStatement = connection.prepareStatement(query);  preparedStatement.setString(1, messageContent);  ResultSet resultSet = preparedStatement.executeQuery();    if (resultSet.next() || messageContent.isEmpty()) {  int count = resultSet.getInt("count");  return count > 0;  }   return false;  } } |

mainApp.fxml

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>  <?*import javafx.scene.control.\**?> <?*import javafx.scene.layout.\**?> <?*import javafx.scene.text.\**?>* <Pane prefHeight="800.0" prefWidth="1200.0" xmlns="http://javafx.com/javafx/11.0.14-internal" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.kiku.formallanguagemachineteaching.BaiesController">  <TableView fx:id="wordsTable" layoutX="75.0" layoutY="75.0" prefHeight="439.0" prefWidth="687.0">  <columns>  <TableColumn fx:id="wordColumn" prefWidth="128.0" text="Words" />  <TableColumn fx:id="spamColumn" prefWidth="157.0" text="Spam" />  <TableColumn fx:id="noSpamColumn" prefWidth="174.0" text="no Spam" />  <TableColumn fx:id="pSpamColumn" prefWidth="103.0" text="P (Spam)" />  <TableColumn fx:id="pNoSpamColumn" prefWidth="124.0" text="P {No Spam}" />  </columns>  <columnResizePolicy>  <TableView fx:constant="CONSTRAINED\_RESIZE\_POLICY" />  </columnResizePolicy>  </TableView>  <TextArea fx:id="messageField" layoutX="75.0" layoutY="528.0" prefHeight="144.0" prefWidth="417.0" promptText="Исследуемое сообщение" wrapText="true">  <font>  <Font size="16.0" />  </font></TextArea>  <TextField fx:id="statusFiled" alignment="CENTER" editable="false" layoutX="297.0" layoutY="726.0" prefHeight="58.0" prefWidth="193.0" promptText="СТАТУС">  <font>  <Font size="20.0" />  </font>  </TextField>  <Button fx:id="calculateButton" layoutX="75.0" layoutY="726.0" mnemonicParsing="false" onAction="#calculate" prefHeight="58.0" prefWidth="209.0" text="Определить">  <font>  <Font name="System Bold" size="16.0" />  </font>  </Button>  <TableView fx:id="messageTable" layoutX="791.0" layoutY="75.0" prefHeight="439.0" prefWidth="390.0">  <columns>  <TableColumn fx:id="messageColumn" maxWidth="15000.0" prefWidth="75.0" text="Message" />  <TableColumn fx:id="statusColumn" maxWidth="75.0" minWidth="75.0" prefWidth="75.0" text="Status" />  </columns>  <columnResizePolicy>  <TableView fx:constant="CONSTRAINED\_RESIZE\_POLICY" />  </columnResizePolicy>  </TableView>  <TextField fx:id="coffField" layoutX="75.0" layoutY="682.0" prefHeight="27.0" prefWidth="50.0" />  <Button fx:id="removeButton" layoutX="1110.0" layoutY="620.0" mnemonicParsing="false" onAction="#removeMessage" prefHeight="50.0" prefWidth="50.0" text="🗑">  <font>  <Font size="15.0" />  </font></Button>  <TextArea fx:id="otherMessageField" layoutX="791.0" layoutY="528.0" prefHeight="144.0" prefWidth="295.0" promptText="Поле для Таблицы Messages" wrapText="true">  <font>  <Font size="16.0" />  </font>  </TextArea>  <Button fx:id="addButton" layoutX="1110.0" layoutY="530.0" mnemonicParsing="false" onAction="#addMessage" prefHeight="50.0" prefWidth="50.0" text="➕">  <font>  <Font size="15.0" />  </font>  </Button>  <CheckBox fx:id="checkBoxSpam" layoutX="791.0" layoutY="686.0" mnemonicParsing="false" text="SPAM?" />  <Label layoutX="135.0" layoutY="686.0" prefWidth="259.0" text="Коэффициент сглаживания 0 &lt; a ≤ 1" />  <TextField fx:id="pSpamField" editable="false" layoutX="600.0" layoutY="530.0" promptText="Вероятность спама" />  <TextField fx:id="pNoSpamField" editable="false" layoutX="600.0" layoutY="587.0" promptText="Вероятность Не спама" />  <Label layoutX="523.0" layoutY="534.0" text="P (&quot;СПАМ&quot;)" />  <Label layoutX="503.0" layoutY="591.0" text="P (&quot;НЕ СПАМ&quot;)" />  <Label fx:id="stavniLabel" layoutX="677.0" layoutY="561.0" text="||" />  <Label layoutX="498.0" layoutY="738.0" prefWidth="90.0" text="Слова исключения" textAlignment="CENTER" wrapText="true" />  <TextField fx:id="clearWordField" layoutX="590.0" layoutY="726.0" prefHeight="58.0" prefWidth="580.0" promptText="Слова-исключения" text="в на из от до по за под над к с у о об перед через между подо и" /> </Pane> |

На рисунке 1 представлен пример работы программы.

**Ввод:**

*Сообщение*: сейчас давай сходим в магазин, я схожу куплю три булки хлеба, а ты купи игральные карты. Будем показывать фокусы и розыгрыш на большой приз.

*Коэффициент сглаживания*: 1

**Вывод:**

*P(“СПАМ”)* = 9.507505702149955E-38

*P(“НЕ СПАМ”)* = 1.6207125771961644E-37

*Статус:* не спам

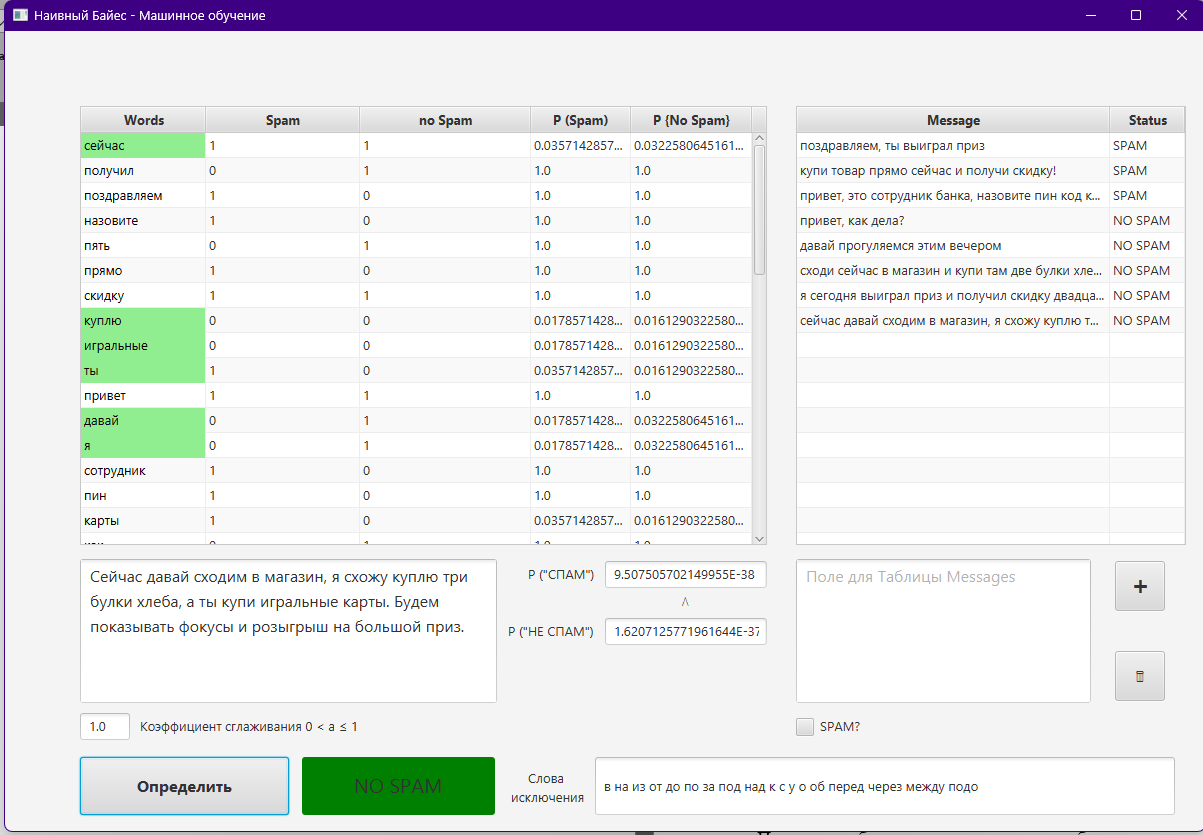


Рисунок 1 – Пример работы программы

Вывод: я научился классифицировать текст по алгоритму наивного Байеса и улучшил опыт программирования на языке Java, освоил основные понятия элементов машинного обучения