Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 5 по дисциплине «Методы машинного обучения»

Тема Наивный байесовский классификатор для фильтрации спама

Студент Сапожков А.М,

Группа ИУ7-23М

Преподаватель Солодовников В.И.

Москва, 2025

Содержание

ВВЕДЕНИЕ **4**

1 Аналитическая часть **5**

1.1 Теорема Байеса 5

1.2 Классификация 5

1.3 Наивный байессовский классификатор 5

2 Технологическая часть **7**

2.1 Средства реализации 7

2.2 Реализация алгоритмов 7

3 Исследовательская часть **11**

3.1 Среда для тестирования 11

ЗАКЛЮЧЕНИЕ **15**

ВВЕДЕНИЕ

В современной теории машинного обучения задача классификации является одной из фундаментальных проблем, имеющей широкий спектр практических приложений. Наивный байесовский классификатор представляет собой важный базовый алгоритм, демонстрирующий ключевые принципы вероятностного подхода к классификации данных.

Целью данной лабораторной работы является изучение принципов функционирования наивного байесовского классификатора.

Задачи данной лабораторной работы:

1) осуществить предобработку данных, сформировать обучающий и тестовый наборы данных;
2) построить наивный байесовский классификатор;
3) оценить качество работы классификатора.

1 Аналитическая часть

1.1 Теорема Байеса

Теорема Байеса (или формула Байеса) — одна из основных теорем элементарной теории вероятностей, которая позволяет определить вероятность какого-либо события при условии, что произошло другое статистически взаимозависимое с ним событие. Другими словами, по формуле Байеса (1.1) можно более точно пересчитать вероятность, взяв в расчёт как ранее известную информацию, так и данные новых наблюдений.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}, \tag{1.1}$$

где

— $P(A)$ — априорная вероятность гипотезы A (prior probability);

— $P(A|B)$ — вероятность гипотезы A при наступлении события B (апостериорная вероятность — posterior probability);

— $P(B|A)$ — вероятность наступления события B при истинности гипотезы A (правдоподобие — likelihood);

— $P(B)$ — полная вероятность наступления события B (вероятность данных evidence).

1.2 Классификация

Классификация (classification) — это задача присвоения меток класса (class label) наблюдениям (Observation) объектам из предметной области. Множество допустимых меток класса конечно. В свою очередь класс — это множество всех объектов с данным значением метки. Требуется построить алгоритм, способный классифицировать (присвоить метку) произвольный объект из исходного множества. Классификация, как правило, на этапе настройки использует обучение с учителем.

1.3 Наивный байессовский классификатор

Широкий класс алгоритмов классификации, основанный на принципе максимума апостериорной вероятности. Для классифицируемого объекта вычисляются функции правдоподобия каждого из классов, по ним вычисляются апостериорные вероятности классов. Байесовский классификатор использует оценку апостериорного максимума (Maximum a posteriori estimation) для определения наиболее вероятного класса. Объект относится к тому классу, для которого апостериорная вероятность максимальна.

Байесовский подход к классификации основан на теореме, утверждающей, что если плотности распределения каждого из классов известны, то искомый алгоритм можно выписать в явном аналитическом виде. Более того, этот алгоритм оптимален, то есть обладает минимальной

вероятностью ошибок.

На практике плотности распределения классов не известны. Их приходится оценивать (восстанавливать) по обучающей выборке. В результате байесовский алгоритм перестаёт быть оптимальным, так как восстановить плотность по выборке можно только с некоторой погрешностью. Чем короче выборка, тем выше шансы подогнать распределение под конкретные данные и столкнуться с эффектом переобучения.

Вероятностная модель для классификатора — это условная модель $P(C|F_1, \ldots, F_n)$ над зависимой переменной класса C с малым количеством результатов или классов, зависимая от нескольких переменных $F_1, \ldots, F_n$.

Используя теорему Байеса, запишем:

$$P(C|F_1, \ldots, F_n) = \frac{P(C)P(F_1, \ldots, F_n|C)}{P(F_1, \ldots, F_n)}. \tag{1.2}$$

На практике интересен лишь числитель этой дроби, так как знаменатель не зависит от C и значения свойств F_i даны, так что знаменатель — константа. Числитель эквивалентен совместной вероятности модели $P(C, F_1, \ldots, F_n))$, которая может быть переписана, используя повторные приложения определений условной вероятности:

$$\begin{aligned} P(C, F_1, \ldots, F_n) &= P(C)P(F_1, \ldots, F_n|C) = \\ &= P(C)P(F_1|C)P(F_2, \ldots, F_n|C, F_1) = \\ &= P(C)P(F_1|C)P(F_2|C, F_1)P(F_3, \ldots, F_n|C, F_1, F_2) = \\ &= P(C)P(F_1|C)P(F_2|C, F_1) \ldots P(F_n, \ldots, F_n|C, F_1, F_2, F_3, \ldots, F_{n-1}). \end{aligned} \tag{1.3}$$

2 Технологическая часть

2.1 Средства реализации

В качестве языка программирования для реализации алгоритмов был выбран язык программирования Python ввиду наличия библиотек для обучения регрессионных моделей, таких как sklearn и numpy.

2.2 Реализация алгоритмов

На листинге 2.1 представлена реализация алгоритма фильтрации спама с использованием наивного байесовского классификатора.

Листинг 2.1 — Фильтрация спама с использованием наивного байесовского классификатора

```
import kagglehub

path = kagglehub.dataset_download("uciml/sms-spam-collection-dataset"
   )

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix,
   matthews_corrcoef, ConfusionMatrixDisplay
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud

nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')

df = pd.read_csv(path+'/spam.csv', encoding='latin-1')
df = df.rename(columns={'v1': 'label', 'v2': 'message'})
df['label'] = df['label'].map({'ham': 0, 'spam': 1})
```

```
def preprocess_text(text):
  text = text.lower()
  text = re.sub(r'[^a-zA-Z\s]', '', text)
  tokens = word_tokenize(text)
  stop_words = set(stopwords.words('english'))
  tokens = [token for token in tokens if token not in stop_words]
  return ' '.join(tokens)

df['processed_message'] = df['message'].apply(preprocess_text)
X = df['processed_message']
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(
  X, y, test_size=0.2, random_state=42, stratify=y
)

pipeline = Pipeline([
('tfidf', TfidfVectorizer(max_features=5000)),
('classifier', MultinomialNB())
])
pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)

print("\nОтчет о классификации:")
print(classification_report(y_test, y_pred, target_names=['Ham', '
   Spam']))
print("\nМатрица ошибок:")
conf_matrix = confusion_matrix(y_test, y_pred)
print(conf_matrix)
cmd = ConfusionMatrixDisplay(conf_matrix, display_labels=['Ham','Spam
   '])
cmd.plot()

tn, fp, fn, tp = conf_matrix.ravel()
accuracy = (tp + tn) / (tp + tn + fp + fn)
precision = tp / (tp + fp)
recall = tp / (tp + fn)
f1 = 2 * (precision * recall) / (precision + recall)
mcc = matthews_corrcoef(y_test, y_pred)
print("\nДополнительные метрики:")
print(f"Точность (Accuracy): {accuracy:.4f}")
```

```
print(f"Точность (Precision): {precision:.4f}")
print(f"Полнота (Recall): {recall:.4f}")
print(f"F1-мера: {f1:.4f}")
print(f"MCC: {mcc:.4f}")

fig, ax = plt.subplots(figsize=(7,7))
sns.barplot(x=y.unique(), y=y.value_counts(), ax=ax)
ax.bar_label(ax.containers[0], fmt='%.i mails', fontsize=12, padding
   =0)
plt.xticks(rotation=45)
plt.xlabel('Message Types')
plt.ylabel('Message Counts')
plt.title('Message Types and Counts')

text_ham=X.where(y==0).str.cat(sep=' ')
text_spam=X.where(y==1).str.cat(sep=' ')
wordcloud_ham = WordCloud(background_color='lightblue').generate(
   text_ham)
wordcloud_spam = WordCloud(background_color='red').generate(text_spam
   )
plt.imshow(wordcloud_ham, interpolation='bilinear')
plt.title('Legitimate Messages', fontdict={'fontsize' : 20, 'color' :
    'blue'})
plt.axis("off")
plt.show()
plt.imshow(wordcloud_spam, interpolation='bilinear')
plt.title('Spam Messages', fontdict={'fontsize' : 20, 'color' : 'red
   '})
plt.axis("off")
plt.show()

print("\nПример классификации новых сообщений:")
test_messages = [
  "URGENT! You have won a 1 week FREE membership in our 100,000 Prize
      Jackpot!",
  "Hey, what time are we meeting for lunch tomorrow?",
  "Congratulations! You've been selected to receive a free iPhone!
     Click here!",
  "Don't forget to bring your laptop to the meeting"
]
predictions = pipeline.predict([preprocess_text(msg) for msg in
```

```
    test_messages])
for msg, pred in zip(test_messages, predictions):
print(f"{'SPAM' if pred == 1 else 'HAM'}: {msg}")
```

3 Исследовательская часть

3.1 Среда для тестирования

Для тестирования разработанного алгоритма применялась облачная платформа Google Colab, не требующая установки ПО на локальный компьютер.

Листинг 3.1 — Отчёт по результатам классификации

```
                precision    recall  f1-score   support

Ham                 0.96      1.00      0.98       966
Spam                0.99      0.74      0.85       149

accuracy                              0.97      1115
macro avg           0.98      0.87      0.92      1115
weighted avg        0.97      0.97      0.96      1115

Дополнительные метрики:
MCC: 0.8421
```

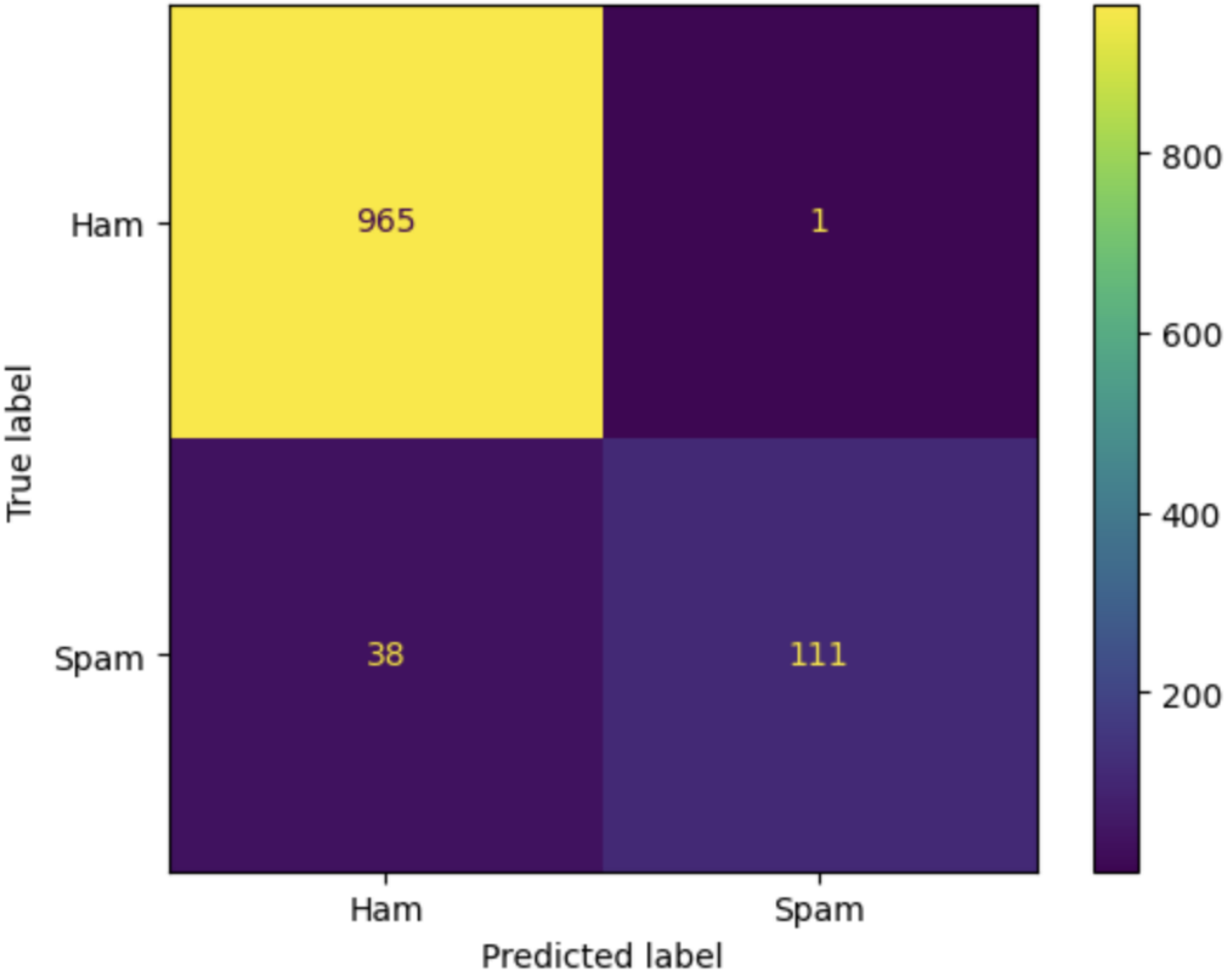

Рисунок 3.1 — Матрица ошибок

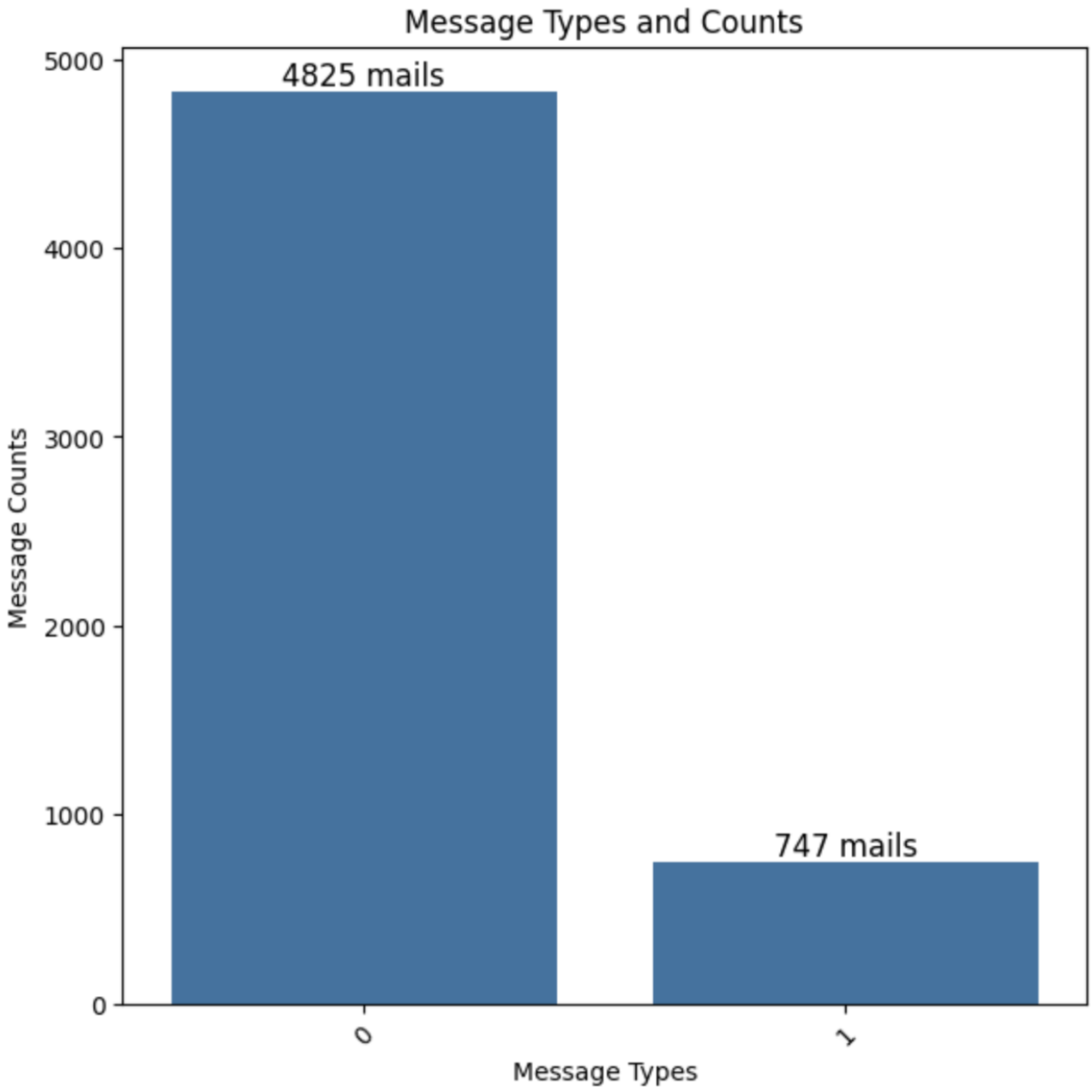

Рисунок 3.2 — Соотношение размеров классов

Рисунок 3.3 — Наиболее часто встречающиеся слова для каждого класса

ЗАКЛЮЧЕНИЕ

В рамках лабораторной работы было проведено изучение принципов функционирования наивного байесовского классификатора.

1. Осуществлена предобработка данных, сформированы обучающий и тестовый наборы данных;
2. Построен наивный байесовский классификатор;
3. Оценено качество работы классификатора.

Для построенного классификатора метрика MCC составила 0.84, в то время как F1-мера — 0.98 для обычных сообщений и 0.85 для спама.