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, искусственный интеллект и системы управления

КАФЕДРА Системы обработки информации и управления

Домашнее задание

по дисциплине «Методы машинного обучения»

Тема: «Классификация тестовых сообщений»

Выполнил: Нобатов И. - студент ИУ5-24М
Проверил: Гапанюк Ю.Г. - к.т.н, доц.

Москва, 2023 г.

1. **ВЫБОР ЗАДАЧИ**

Задачей является изучение способов классификации текстовых сообщений методами машинного обучения в контексте возможности автоматизации систем Service Desk.

2. **ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

Задачи классификации текста изучаются и реализовываются в программном виде уже в течение нескольких десятилетий. Интерес к применению классификаторов текста вырос в особенности с прорывами в области обработки естественного языка, которые произошли во второй половине 2010-ых годов. В большинстве систем классификации текста реализованы четыре основные фазы: извлечение признаков, сокращение размерности, выбор классификатора и оценка модели. [1]

Существует два основных подхода к выделению признаков из текста.
Первый подход основан на «взвешивании слов», то есть на преобразовании слова или иной обрабатываемой единице в такое скалярное значение, которое отражает частоту его использования в тексте. Наиболее активно используемыми техниками из этой группы являются «Weighted Words» и «TF-IDF», который присваивает слову значение пропорционально его использованию в отдельном тексте и обратно пропорционально использованию в остальных текстах пакета, что должно уменьшить влияние частоты использования слова в языке на выделение признаков.

Второй подход основан на векторизации слов и способны лучше учитывать контекст использования слова. В методах векторизации(GloVe, FastText, Word2Vec) могут использоваться классические алгоритмы машинного обучения, а так же различные нейросетевые модели.

Для снижения размерности используются также как классические методы машинного обучения, так и новые методы основанные на нейросетях — автоэнкодерах. Используются такие методы как PCA(метод главных компонент), LDA (линейный дискриминантный анализ), автоэнкодеры, T-SNE и т.д.

Для задачи классификации также используются как методы классического машинного обучения, такие как алгоритм Роккио, логистическая регрессия, наивный байес, так и нейросети (рекуррентные, сверточные, автоэнкодеры).

Поскольку задача классификации заявок в Service Desk является довольно специфической, существует относительно не много исследований на эту тему. Часть из них посвящена изучению применимости классических методов машинного обучения. Рассмотриv их подробнее.

В статье «A machine learning based help desk system for IT service management»[2] за авторством Фераса Аль-Хавари и Халы Бахрам опубликованной в 2019 году изучается ряд моделей построенных с использованием ПО Weka, предназначенных для классификации поступающих заявок. На этапе выделения признаков во всех случаях используется модель векторного представления, а на этапе классификации каждая модель обучалась с использованием одного из классических алгоритмов машинного обучения, таких как SVM, Naïve Bayes, Rule-based и Decision Tree. Наилучшим образом себя показал алгоритм SVM-алгоритм на основе SMO. Изначально авторам удалось добиться точности в 53,8%, затем после применения предобработки и выделения в отдельные наборы заголовков и описаний 81,4%. Результаты тестирования моделей приведены на рисунке 1 и рисунке 2.

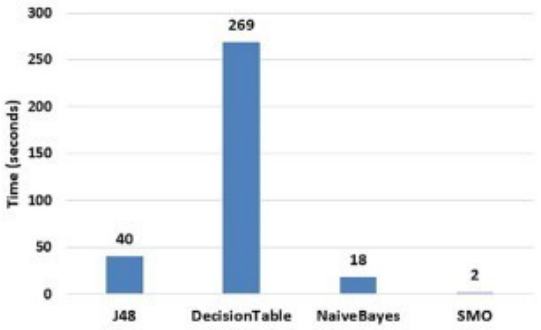


Fig. 21. The elapsed time to build and test the classification models using the four machine learning algorithms.

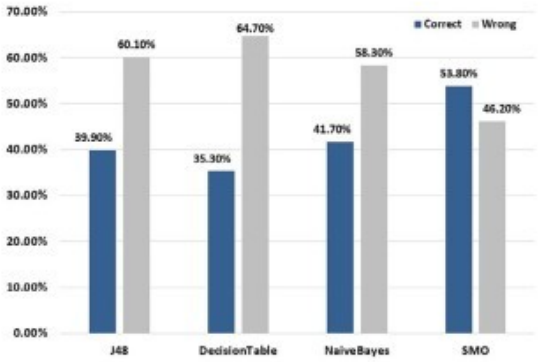


Fig. 22. The prediction accuracy of the four generated models using the default feature vectorization parameters.

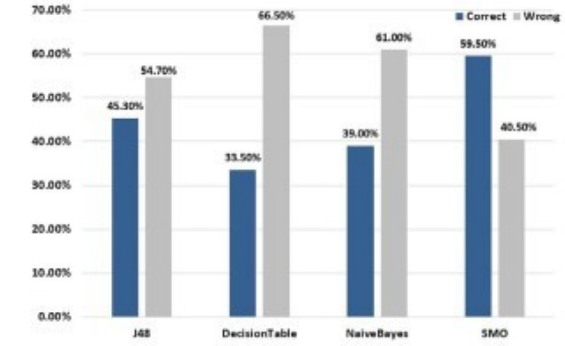


Fig. 23. The prediction accuracy of the four generated models when the TF-IDF feature vector and Lovins stemmer were enabled.

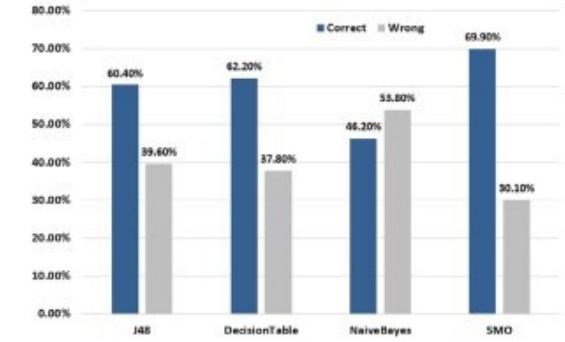


Fig. 24. The effect of the ticket text preprocessing on the prediction accuracy of the four generated models.

Рисунок 1 — Изначальные результаты обучения

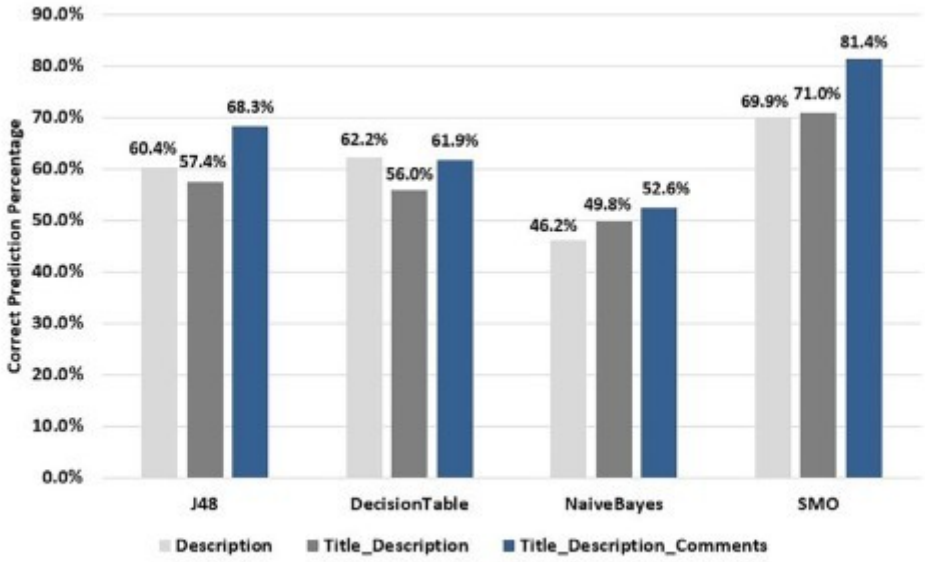


Fig. 25. The effect of considering the ticket title and comments in addition to its description on the prediction accuracy.

Рисунок 2 — Результаты после выделения названия и описания в отдельные наборы

Можно заметить, что авторы не смогли добиться высокой точности на основном наборе данных. Поскольку даже результат точности в 81,4% был достигнут лишь на отдельно выделенном датасете с названиями запросов.

Рассмотрим исследование выполненное Microsoft в сотрудничестве с Envada.[3] Исследование также выполнялось с использованием специального ПО (Azure Machine Learning Studio, Microsoft Cognitive Toolkit, Azure Machine Learning service), его целью ставилось создание модели для автоматической классификации технических заявок для службы поддержки Endava. В качестве набора данных использовались внутренние данные Envada (50 тысяч классифицированных технических заявок с оригинальными сообщениями от пользователей и уже назначенными метками). Данные подвергались значительной предобработке и кодированию конфиденциальных данных, однако в рамках предобработки не была устранена несбалансированность набора данных, заключающаяся в том, что для некоторых классов заявок в датасете было слишком мало данных, что серьезно повлияло на результаты построения модели. То, как представлены текстовые сообщения, можно увидеть на рисунке 3.

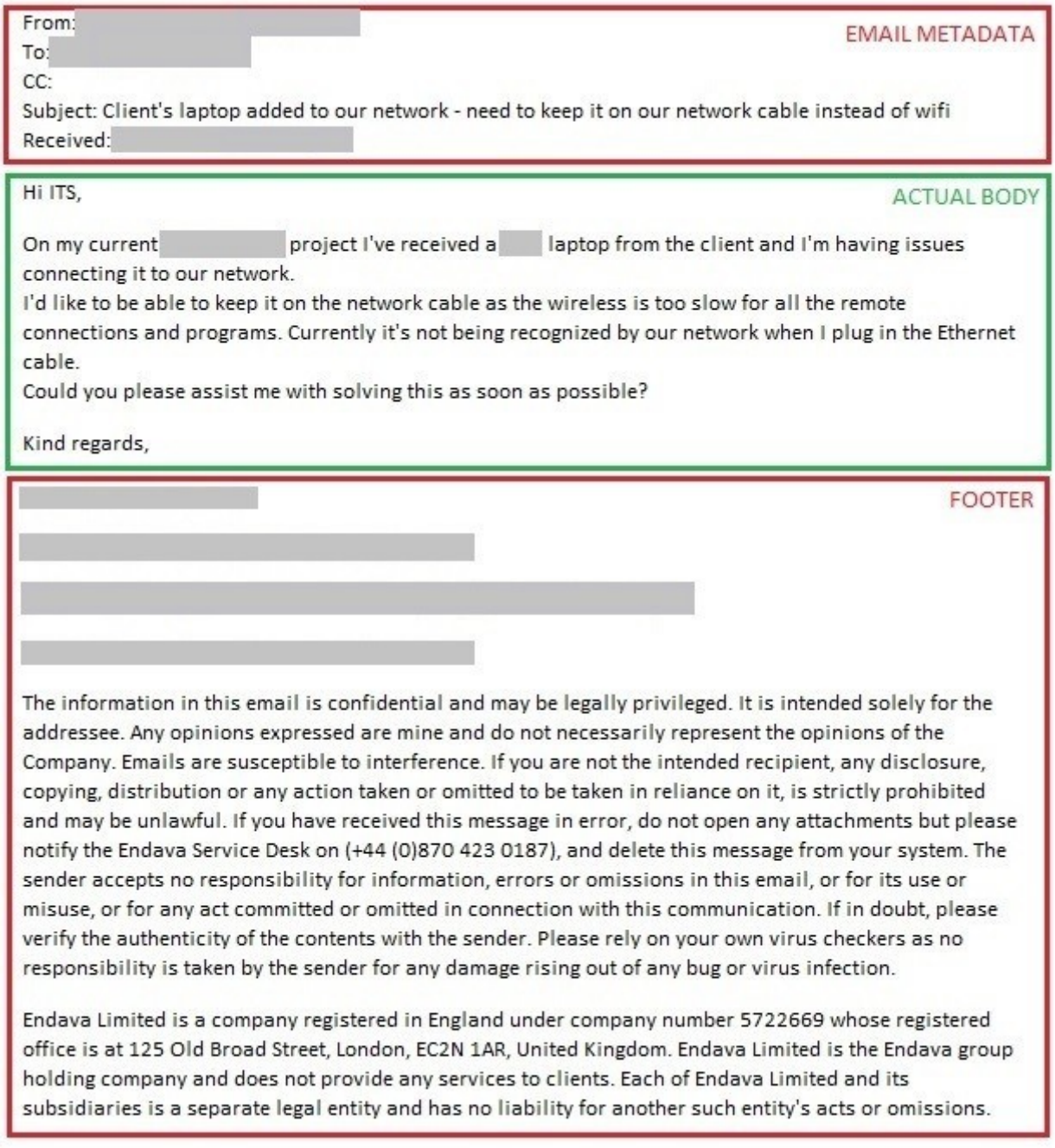


Рисунок 3 — Представление текстовых сообщений

На рисунке 4 можно увидеть представление данных после обработки. На рисунке 5 продемонстрирована последовательность действий для обработки данных.

all_tickets

Columns: 8
Rows: 48682

	body	ticket_type	category	sub_category1	sub_category2	business_service	urgency	impact
1	hi since recruiter lead permission approve requis...	1	4	2	21	71	2	3
2	connection with icon icon dear please setup ico...	1	6	22	7	26	2	3
3	work experience user work experience user hi w...	1	5	13	7	32	2	3
4	requesting for meeting requesting meeting hi pl...	1	5	13	7	32	2	3
5	reset passwords for external accounts re expire ...	1	4	2	76	4	2	3
6	mail verification warning hi has got attached ple...	1	4	3	7	89	2	3
7	mail please dear looks blacklisted receiving mail...	1	4	3	87	89	2	3

Рисунок 4 — Представление данных после обработки

Рисунок 4 — Шаги при обработке

Несбалансированность данных продемонстрирована на рисунке 5.

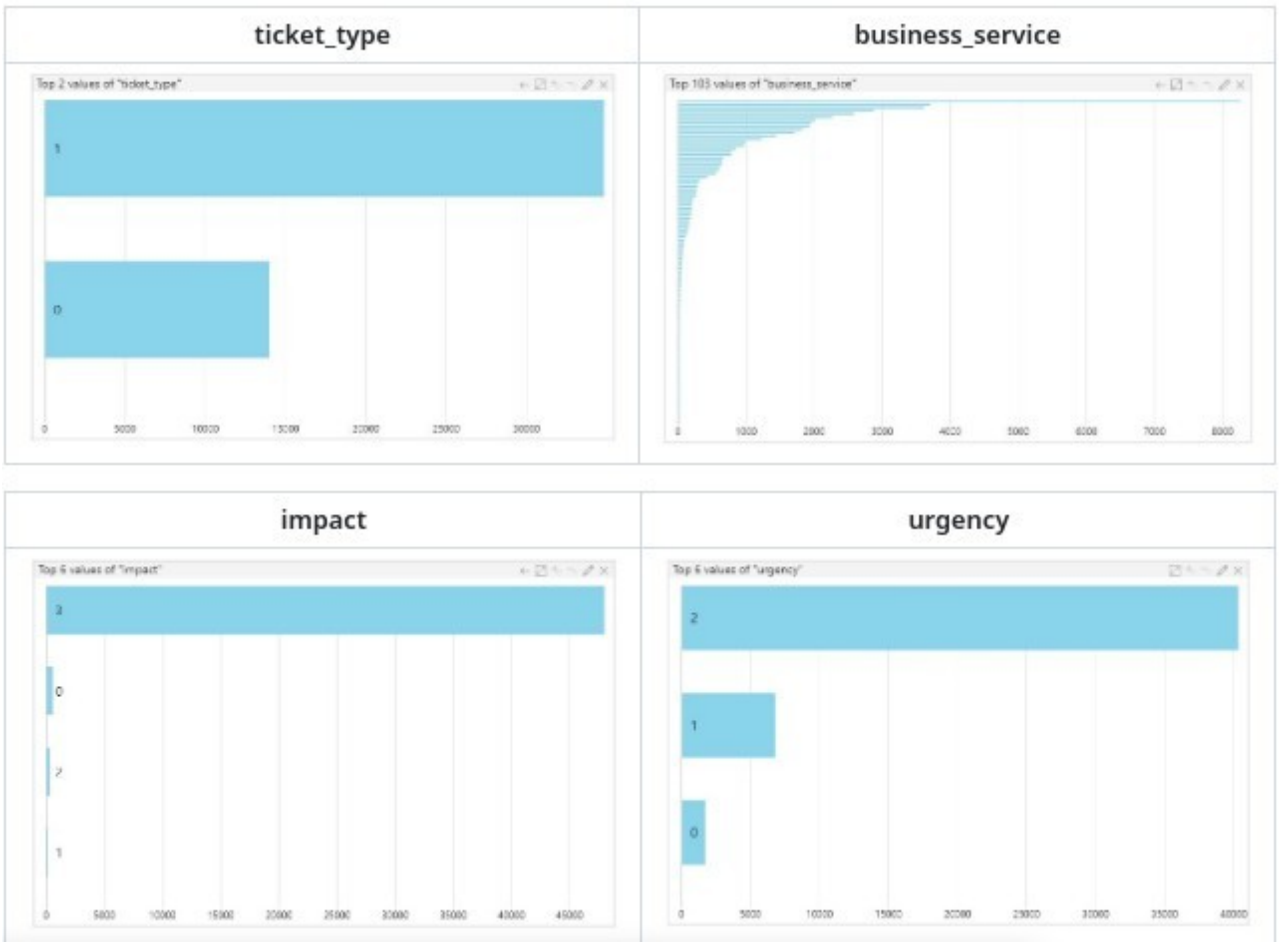
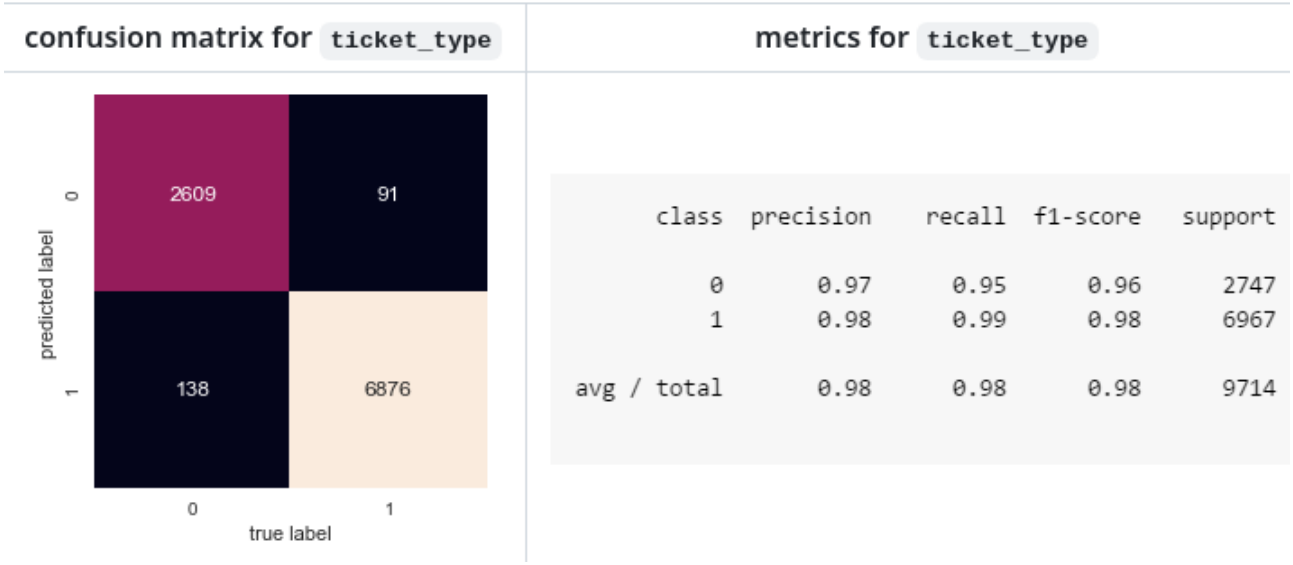


Рисунок 5 — Несбалансированность данных по некоторым столбцам

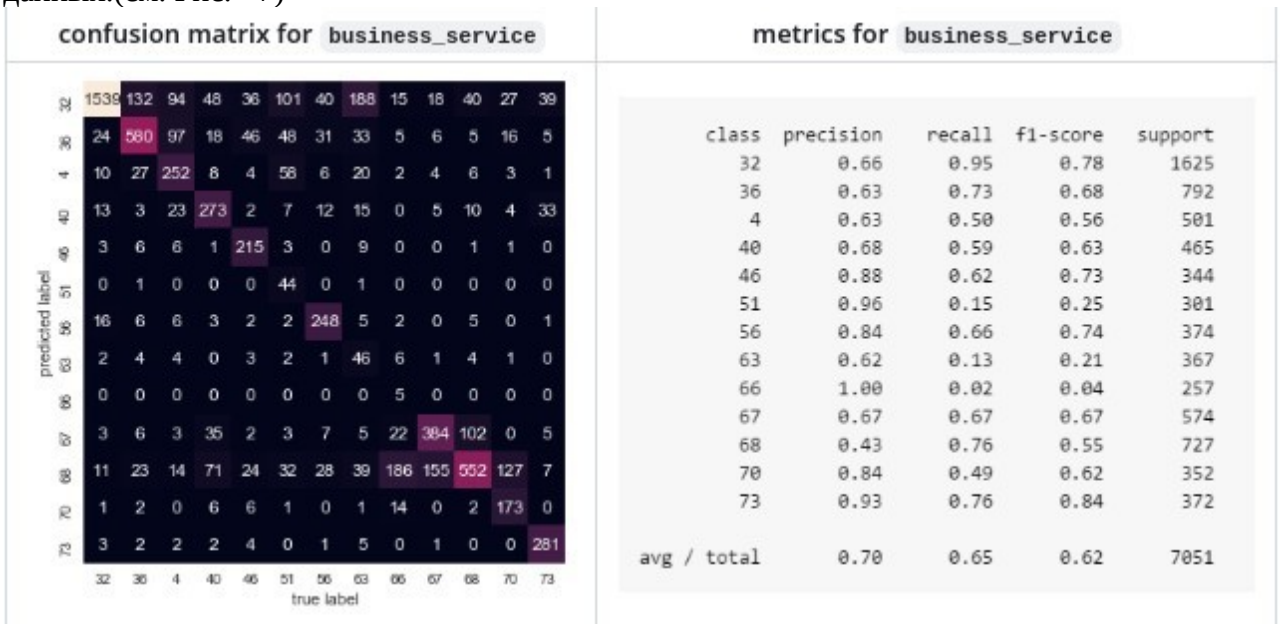
Для классификации были использованы 2 алгоритма классического машинного: SVM и наивный Байес. В обоих случаях результаты были довольно похожи, но для некоторых моделей наивный Байес показал значительно лучшие результаты. При обучении модели на наименее несбалансированном столбце (ticket_type) были достигнуты наилучшие результаты, которые приведены на рисунке 6.



class	precision	recall	f1-score	support
0	0.97	0.95	0.96	2747
1	0.98	0.99	0.98	6967
avg / total	0.98	0.98	0.98	9714

Рисунок 6 — Результаты обучения на первом этапе

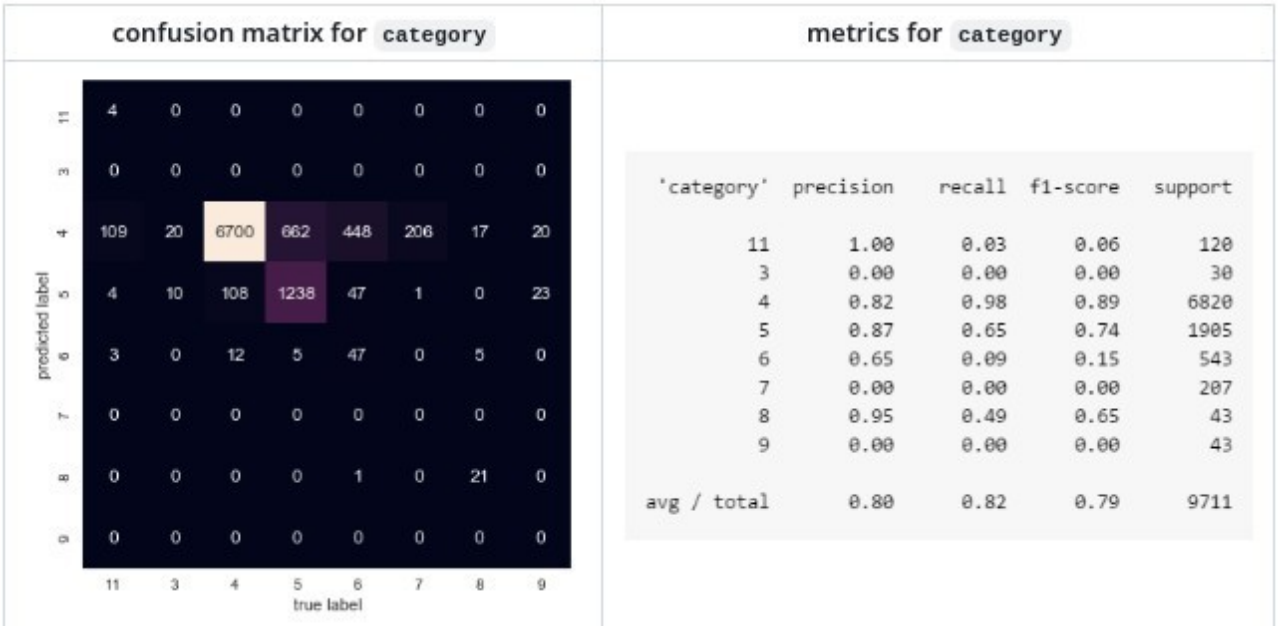
Относительно приемлемые результаты были достигнуты для столбца business_service. Для этого пришлось отбросить значительную часть значений с недостаточным количеством данных.(см. Рис. - 7)



class	precision	recall	f1-score	support
32	0.66	0.95	0.78	1625
36	0.63	0.73	0.68	792
4	0.63	0.50	0.56	501
40	0.68	0.59	0.63	465
46	0.88	0.62	0.73	344
51	0.96	0.15	0.25	301
56	0.84	0.66	0.74	374
63	0.62	0.13	0.21	367
66	1.00	0.02	0.04	257
67	0.67	0.67	0.67	574
68	0.43	0.76	0.55	727
70	0.84	0.49	0.62	352
73	0.93	0.76	0.84	372
avg / total	0.70	0.65	0.62	7051

Рисунок 7 — Результаты обучения на втором этапе

Для колонок `category`, `impact` and `urgency` не удалось достичь примелемых результатов из-за сильной несбалансированности. (см. Рис. - 8,9,10)



'category'	precision	recall	f1-score	support
11	1.00	0.03	0.06	120
3	0.00	0.00	0.00	30
4	0.82	0.98	0.89	6820
5	0.87	0.65	0.74	1905
6	0.65	0.09	0.15	543
7	0.00	0.00	0.00	207
8	0.95	0.49	0.65	43
9	0.00	0.00	0.00	43
avg / total	0.80	0.82	0.79	9711

Рисунок 8 — Результаты обучения на третьем этапе

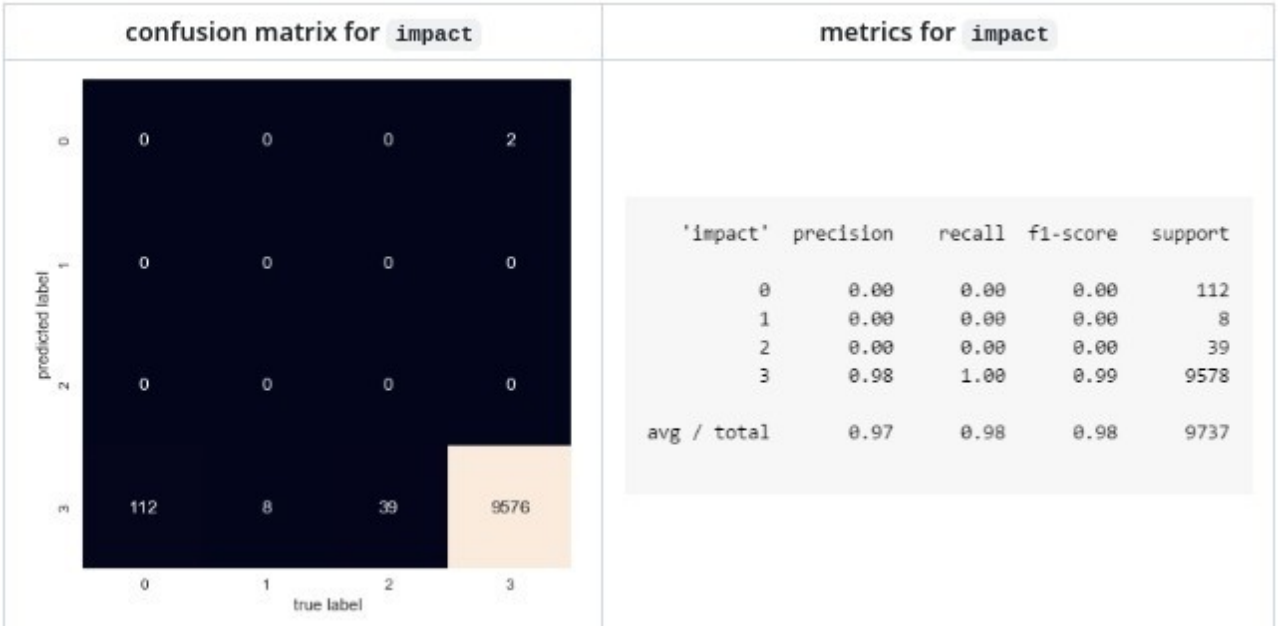


Рисунок 9 — Результаты обучения на четвертом этапе

Рисунок 10 — Результаты обучения на пятом этапе

Датасет и исходный код, используемый в данном исследовании находится в открытом доступе. Разработчики в рамках исследования осуществили разработку веб-сервиса для предоставления доступа к модели. Исходники ПО также доступны в репозитории.

Рассмотрим еще одно исследование «IT Ticket Classification», представляющее интерес в контексте данной работы, опубликованное на сайте «analyticsinsight.net».[4]

В данной работе ставилась цель построения модели, способной осуществлять классификацию на трех уровнях. Для обучения использовался соответствующий датасет. Распределение данных по категориям отображено на рисунках 11, 12, 13.

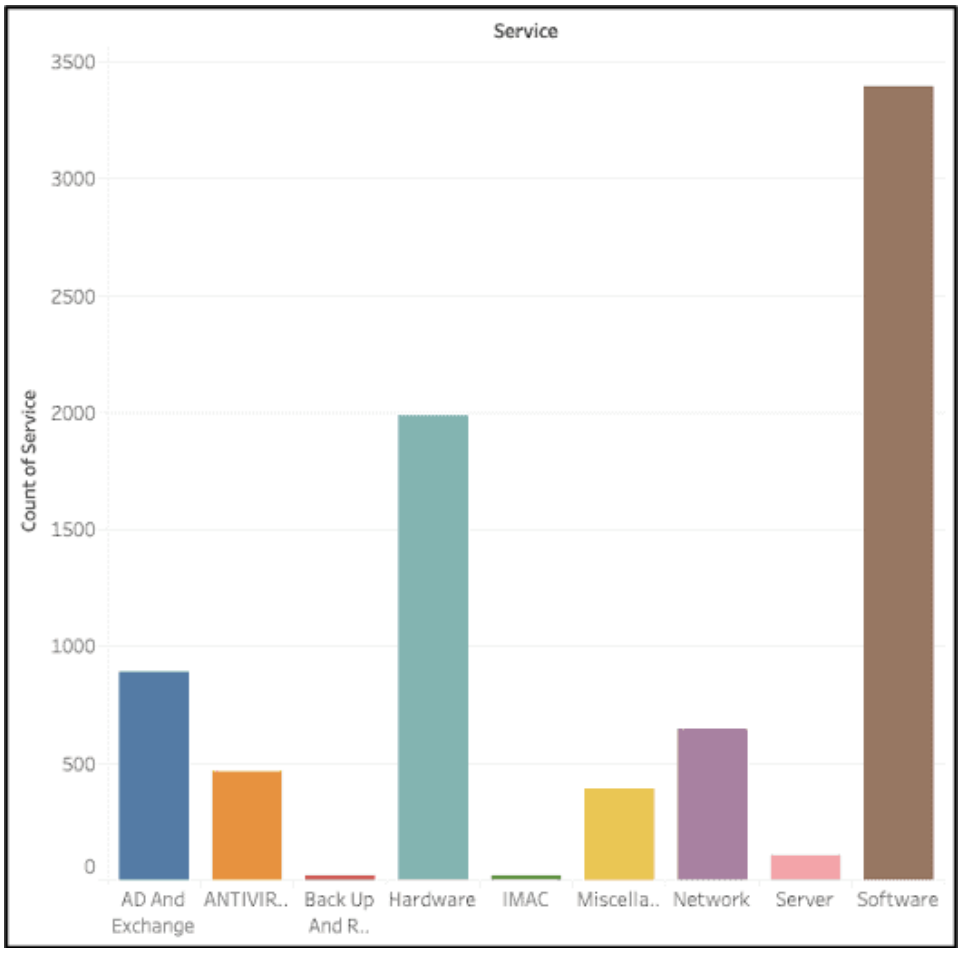


Рисунок 11 — Распределение по колонке «Service»

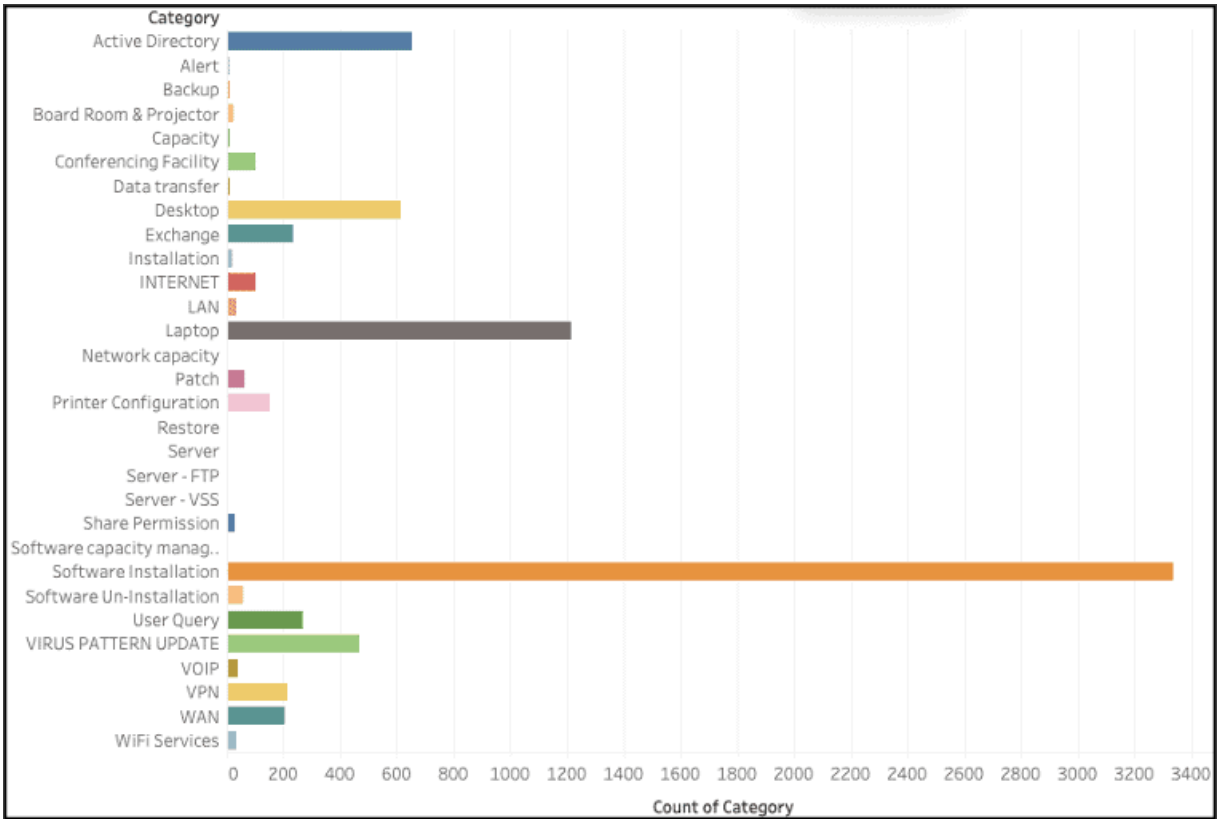


Рисунок 12 — Распределение по колонке «Category»

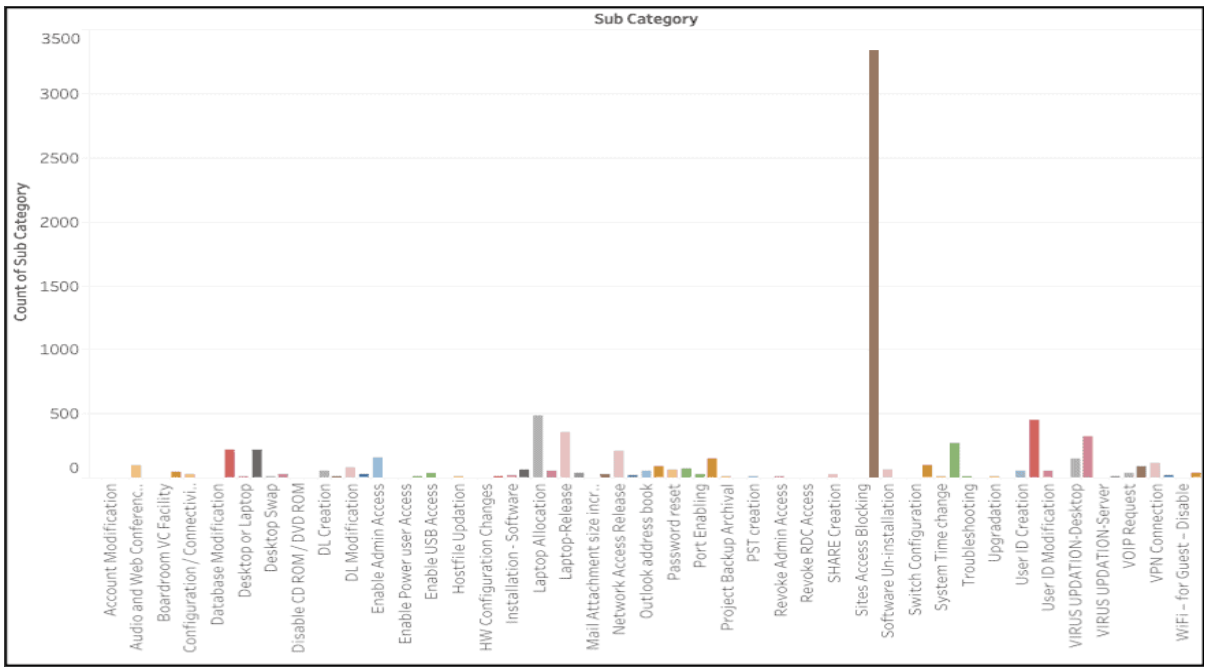


Рисунок 13 — Распределение по колонке «Sub Category»

Структура вложенности классов отображена на рисунке 14.

TIER 1 (SERVICE)	TIER 2 (CATEGORY)	TIER 3 (SUB CATEGORY)
Hardware	Laptop	Laptop Allocation
		Laptop- Release
		Provide Access (Admin/ USB/Encryption/RDC/Power user)
		Disable CD/DVD ROM
		Laptop upgrade
		Revoke admin/RDC access
		Laptop swap
	Desktop	System Movement
		Desktop- Release
		Desktop Allocation
		Desktop Release
		Desktop Swap
		Desktop upgrade
		Enable access (admin/host file/USB/RDC)
		Disable/Revoke access (RDC/Admin/Telnet)
	Server	Configuration
		Server Upgrade
Ad And Exchange	Active Directory	User ID (Modification/ Creation/Deletion)
		Password Reset
		Domain change
		OU Movements
		Hostname deletion
		Update designation
	Exchange	DL (Modification/ Creation/ Deletion)
		Outlook Address Book
		Mailbox Movement/Mail attachment size increase
		PST creation
Network	VPN	Connection
		Account Creation
		Password Reset
	WAN	Network Access Release
		Sites Access Blocking
	Internet	Access
		Configuration
		Revoke internet access
	LAN	Port Enabling
		Subnet change
		Switch configuration
		VLAN Configuration
	Network Capacity	Upgrade
	VOIP	VOIP Request
	Wifi	Wi fi for guest (enable/disable)
Anti-Virus	Virus-Pattern Update	Laptop
		Desktop
		Server

Рисунок 14 — Структура вложенности классов

Из исходных данных были выбраны только те столбцы, где по крайней мере 70% данных не были помечены как "NA". На выбранных столбцах были применены методы отбора признаков, чтобы определить, какие из них являются полезными. Поскольку независимые и зависимые переменные были категориальными, для понимания зависимости был использован тест хи-квадрат. Был использован алгоритм "SelectKBest" с методом хи-квадрат:

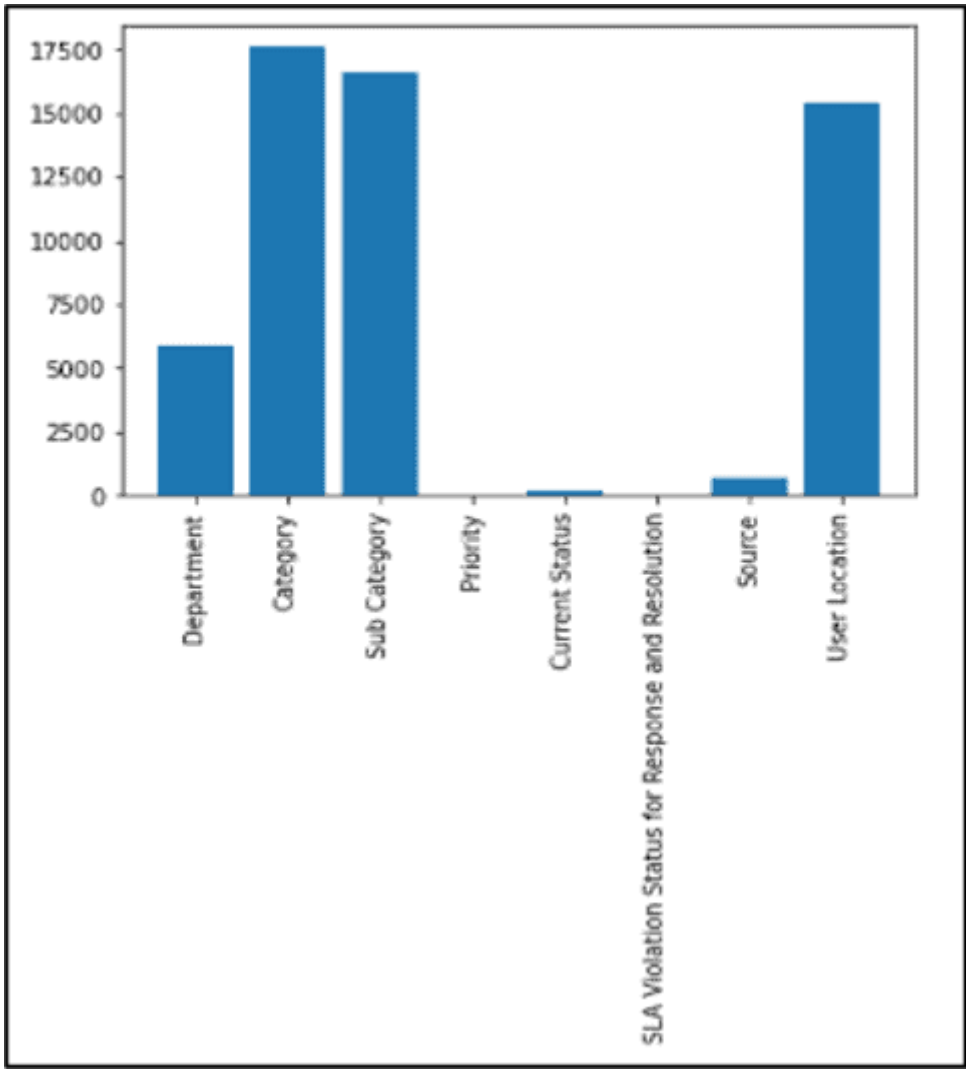


Рисунок 15 — Результаты

Ось Y представляет собой статистику хи-квадрат. Чем выше этот показатель, тем больше зависимость "Сервиса" от конкретной переменной. Только 'Отдел', 'Категория', 'Подкатегория' и 'Местоположение пользователя' имели значительный уровень зависимости с первым уровнем иерархии: 'Сервис'. Поскольку 'Категория' и 'Подкатегория' являются уровнями последующей иерархии, они не будут учитываться при построении модели.
На этапе выделения признаков авторы использовали различные методы. Результаты приведены ниже на рисунке 16.

Embeddings	F1 Scores	Classification Model
BERT	94.71	XGBoost
ELMO	95.21	XGBoost
Spacy	93.21	Gradient Boosting
Tf-Idf	93.48	XGBoost
Doc2Vec	38.32	LightGBM
InferSent	96.50	Gradient Boosting
USE	95.63	Gradient Boosting

Рисунок 16 — Результаты использования различных алгоритмов выделения признаков

При использовании InferSent были достигнуты наилучшие результаты.(рис. - 17)

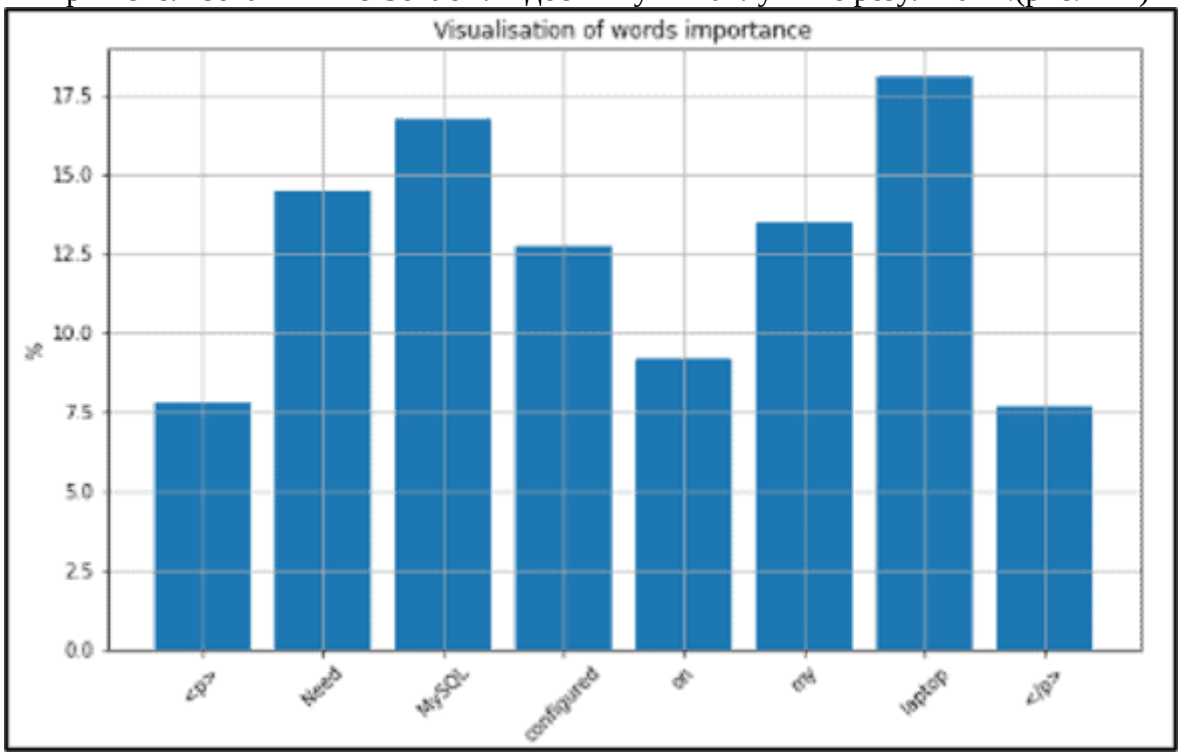


Рисунок 17 — Значимость слов при InferSent

Данные визуализации дают представление о том, насколько хорошо нейросеть InferSent удаётся распознавать ключевые слова в предложении. Таким образом, вышеуказанные визуализации показывают, что InferSent хорошо справляется с пониманием того, что «MySQL» и «Laptop» более важны, чем «Need», «on» или «my». Из визуализации можно также отметить еще один важный момент - что общие слова, используемые, например, «on», «my», имеют очень маленькую важность.

Поэтому метод встраивания становится довольно надежным, так как для текстовых данных не требуется традиционный этап предварительной обработки данных, такой как токенизация, лемматизация, удаление стоп-слов и т.д. Текстовые данные были переданы как есть, и метод работает весьма хорошо.

После этого были опробованы комбинации различных методов встраивания, чтобы увидеть, дадут ли ансамбли методов лучшие результаты. В итоге авторы остановились на следующем:

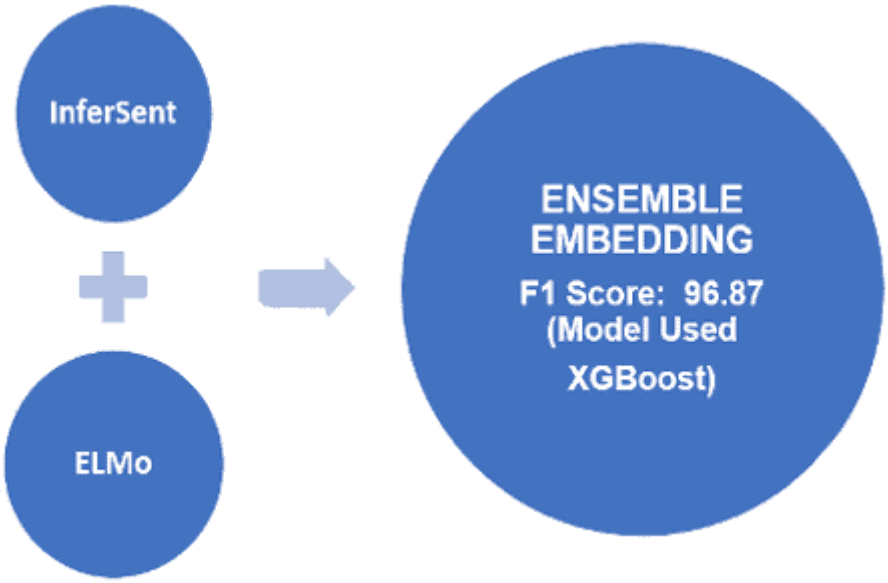


Рисунок 18 — Использованный вариант векторизации

```
def get_infersent_embeddings(x):
    sentences = x
    infersent.build_vocab(sentences, tokenize=True)
    embeddings = infersent.encode(sentences, tokenize=True)
    return embeddings

def elmo_vectors(x):
    graph = tf.Graph()
    with tf.Session(config=config,graph = graph) as sess:
        elmo = hub.Module("https://tfhub.dev/google/elmo/2",trainable=True)
        embeddings = elmo(x.tolist(), signature="default", as_dict=True)["elmo"]
        sess.run(tf.global_variables_initializer())
        sess.run(tf.tables_initializer())
        return sess.run(tf.reduce_mean(embeddings,1))

def get_elmo_embeddings(x):
    train = x
    list_train = [train[i:i+100] for i in range(0,train.shape[0],100)]
    elmo_train = [elmo_vectors(x) for x in list_train]
    elmo_train_new = np.concatenate(elmo_train, axis = 0)
    return elmo_train_new

def ensemble_embedding(x):
    infersent= get_infersent_embeddings(x)
    elmo= get_elmo_embeddings(np.array(x))
    embeddings = np.append(infersent,elmo,axis=1)
    return embeddings
```

Рисунок 19 — Код реализации векторизации

Для уменьшения размерности, существенно выросших после векторизации данных, использовался метод главных компонент.

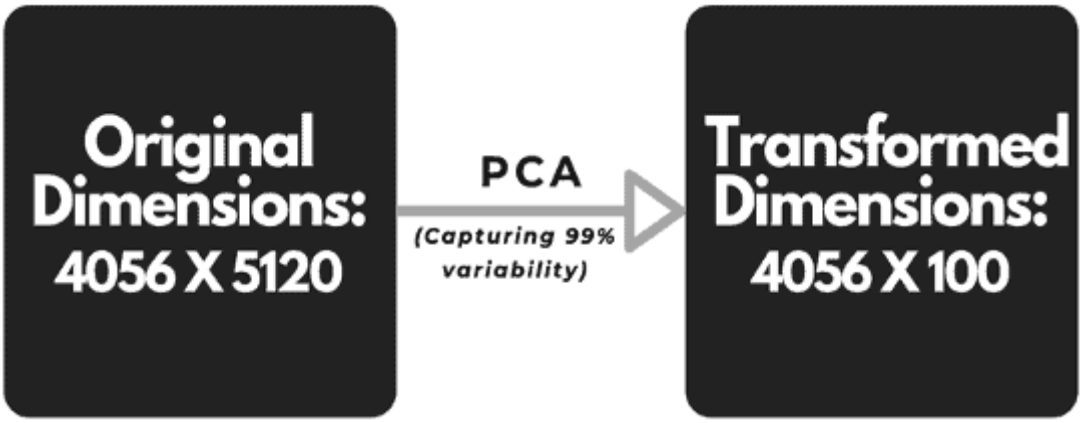


Рисунок 20 — Метод главных компонент для уменьшения размерности

Лучшая модель классификации была выбрана из нескольких алгоритмов машинного обучения на основе производительности на тестовых данных (не видимых для модели).

Модель XgBoost (экстремальный градиентный бустинг) лучше всего справилась с каждым уровнем, и код выглядел следующим образом:

```
def fit_xgb(X_train, X_test, y_train, y_test):
    print("Fitting XG boost Classifier on Data to Calculate accuracy, Fscore and Confusion matrix..")
    xgb = XGBClassifier(colsample_bytree=0.4603, gamma=0.0468, learning_rate=0.05, max_depth=3,
                        min_child_weight=1.7817, n_estimators=2200, reg_alpha=0.4640,
                        reg_lambda=0.8571, subsample=0.5213, silent=1, random_state =7, nthread = -1)
    model = xgb.fit(X_train, y_train)
    print("Test Scores..\n")=
    predictions = model.predict(X_test)
    display('F1 score: '+str(f1_score(y_test,predictions, average='weighted')))
    display('Accuracy: '+str(accuracy_score(y_test,predictions)))
    display(confusion_matrix(y_test,predictions))
    print("\n\nTrain Scores.. \n")
    predictions = model.predict(X_train)
    display('F1 score: '+str(f1_score(y_train,predictions, average='weighted')))
    display('Accuracy: '+str(accuracy_score(y_train,predictions)))
    display(confusion_matrix(y_train,predictions))
    print("\n\nCross validation Scores.. \n")
    scores = cross_val_score(estimator=model, X=X_train, y=y_train, cv=3,
                             scoring = make_scorer(f1_score,average = 'weighted'))
    print(scores)
    print(np.array(scores).sum()/3)
```

Рисунок 21 — Экстремальный градиентный бустинг

Были достигнуты довольно высоки результаты.

Hierarchical Label:	Training F1 Scores:	Cross-Validation F1 Scores:				Testing F1 Scores:
		Training: 1	Training: 2	Training: 3	Average:	
Tier 1	99.999	96.094	94.757	95.236	95.362	96.04
Tier 1+ Tier 2	99.989	91.343	91.125	93.518	91.995	93.26
Tier 1 + Tier 2+ Tier 3	98.889	84.627	83.357	85.605	84.530	83.55

Рисунок 22 — Показатели метрик модели

Стоит отметить, что в данном случае данные так же, как и в предыдущем рассмотренном исследовании, были сильно несбалансированы. Однако полученные результаты явно превосходят результаты предыдущей модели. По-видимому это может быть связано с нейросетевой векторизацией текста и возможно применением экстремального градиентного бустинга, который показывает себя гораздо лучше на несбалансированных данных.

Однако данная модель в качестве признака для классификации использует не только текстовое сообщение, но также данные о локации пользователя и данные о отделе компании, что в определенной степени усложняет сравнение модели с предыдущей.

Теперь рассмотрим статью «Ticket-BERT: Labeling Incident Management Tickets with Language Models» написанную группой сотредников Microsoft.[5] В данной работе решался вопрос построения модели Ticket-BERT на основе нейросети BERT для классификации заявок в системе Service Desk. При это ставилась задача обучить классификатор способный различать одинаково как заявки, сгенерированные пользователем, так и заявки созданные самой системой по определенному шаблону. Также рассматривался вопрос классификации гибридных заявок созданных системой с помощью человека. Для обучения модели были использованы данные разбитые на три набора соответственно (D-Human, D-Machine, D-Mixture). Для векторизации были использованы функции TD-IDF и BoW в базовых моделях. В качестве входных данных в модель использовалась модель Ticket-BERT.

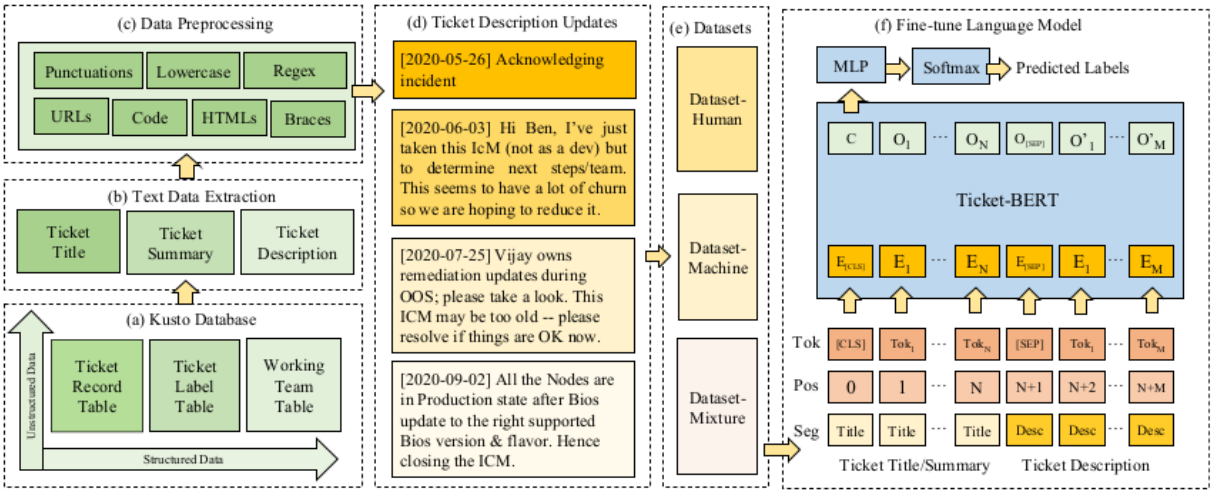


Рисунок 23 — Архитектура модели Ticket-BERT

В качестве классификатора в модели использовалась предобученная модель нейронной сети BERT, которая представляет из себя многослойный двунаправленный трансформер, состоящий из блоков слоев-кодировщиков и декодировщиков и последнего добавляемого слоя полносвязного перцептрона, которой выбирается в зависимости от задачи. При обучении и дообучении модели используется метод маскирования слов.

В рассматриваемом исследовании модель BERT дообучалась на датасете для классификации. Полученные результаты метрик можно посмотреть на рисунке 24.

Dataset	Models	Precision	Recall	F1-Score	AUC
D-Human	NB-BoW	71.07	73.74	69.87	0.955
	NB-TF-IDF	78.18	76.67	76.74	0.973
	LS-Model	84.36	84.08	83.96	**
	LR-BoW	85.16	84.12	84.58	0.98
	LR-TF-IDF	85.81	82.89	83.98	0.986
	Ticket-BERT	**86.40**	**85.50**	**85.90**	**0.988**
D-Machine	NB-BoW	89.17	91.98	89.71	0.997
	NB-TF-IDF	92.28	93.72	92.69	0.999
	LS-Model	96.01	94.52	95.14	**
	LR-BoW	97.91	**98.07**	97.98	1.000
	LR-TF-IDF	97.15	97.50	97.30	1.000
	Ticket-BERT	**98.42**	97.98	**98.19**	**1.000**
D-Mixture	NB-BoW	88.25	91.48	88.88	0.996
	NB-TF-IDF	92.79	94.85	93.61	0.999
	LS-Model	97.24	96.71	96.91	**
	LR-BoW	98.02	**97.93**	97.97	0.999
	LR-TF-IDF	97.61	97.57	97.58	1.000
	Ticket-BERT	**98.62**	97.91	**98.24**	**1.000**

Рисунок 24 — Результаты для датасетов

Можно заметить, что на датасете с данными, сгенерированными машиной, результаты выше, что может являться следствием более строгой структурированности. Далее авторы

исследования выделили заголовок сообщения в отдельный признак для повышения точности модели. Благодаря чему были достигнуты гораздо более высокие результаты на тестовой выборке.

Dataset	Models	Precision	Recall	F1-Score	AUC
D-Human +Title	NB-BoW	91.14	94.13	92.09	0.993
	NB-TF-IDF	93.46	94.23	93.77	0.997
	LS-Model	95.56	97.76	96.49	**
	LR-BoW	97.92	97.90	97.91	0.999
	LR-TF-IDF	97.80	97.15	97.46	0.999
	Ticket-BERT	**98.76**	**99.17**	**98.96**	**1.000**
D-Human +Title +Summary	NB-BoW	92.01	94.69	92.82	0.993
	NB-TF-IDF	94.30	95.03	94.61	0.998
	LS-Model	97.81	97.75	97.77	**
	LR-BoW	98.24	98.15	98.19	0.998
	LR-TF-IDF	97.89	97.33	97.60	0.999
	Ticket-BERT	**98.36**	**98.88**	**98.61**	**1.000**
D-Machine +Title	NB-BoW	91.81	94.04	92.27	0.998
	NB-TF-IDF	94.73	96.10	95.16	0.999
	LS-Model	98.98	99.33	99.13	**
	LR-BoW	98.96	99.23	99.09	1.000
	LR-TF-IDF	98.43	98.87	98.64	1.000
	Ticket-BERT	**99.31**	**99.24**	**99.27**	**1.000**
D-Machine +Title +Summary	NB-BoW	93.16	94.86	93.47	0.999
	NB-TF-IDF	95.08	96.84	95.70	1.000
	LS-Model	98.97	99.29	99.11	**
	LR-BoW	99.20	99.20	99.20	1.000
	LR-TF-IDF	98.92	99.02	98.97	1.000
	Ticket-BERT	**99.34**	**99.35**	**99.35**	**1.000**
D-Mixture +Title	NB-BoW	91.87	94.56	92.60	0.998
	NB-TF-IDF	95.76	97.30	96.43	0.999
	LS-Model	98.07	97.82	97.91	**
	LR-BoW	98.93	98.98	98.95	1.000
	LR-TF-IDF	98.67	98.78	98.72	1.000
	Ticket-BERT	**99.31**	**99.38**	**99.34**	**1.000**
D-Mixture +Title +Summary	NB-BoW	93.29	95.52	93.98	0.998
	NB-TF-IDF	95.51	97.45	96.34	0.999
	LS-Model	98.24	97.03	98.06	**
	LR-BoW	99.09	98.94	99.01	1.000
	LR-TF-IDF	98.75	98.67	98.71	1.000
	Ticket-BERT	**99.37**	**99.34**	**99.36**	**1.000**

Рисунок 24 — Результаты для датасетов

2. Практическая часть

```
import re
from pathlib import Path
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this
version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"
```

```
import torch
from torchtext.vocab import build_vocab_from_iterator
!pip install transformers
from transformers import BertTokenizer
```

```
Requirement already satisfied: transformers in
/opt/conda/lib/python3.10/site-packages (4.28.1)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in
/opt/conda/lib/python3.10/site-packages (from transformers) (0.13.3)
Requirement already satisfied: requests in
/opt/conda/lib/python3.10/site-packages (from transformers) (2.28.2)
Requirement already satisfied: tqdm>=4.27 in
/opt/conda/lib/python3.10/site-packages (from transformers) (4.64.1)
Requirement already satisfied: filelock in
/opt/conda/lib/python3.10/site-packages (from transformers) (3.11.0)
Requirement already satisfied: pyyaml>=5.1 in
/opt/conda/lib/python3.10/site-packages (from transformers) (6.0)
Requirement already satisfied: regex!=2019.12.17 in
/opt/conda/lib/python3.10/site-packages (from transformers)
(2023.3.23)
Requirement already satisfied: numpy>=1.17 in
/opt/conda/lib/python3.10/site-packages (from transformers) (1.23.5)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.10/site-packages (from transformers) (21.3)
Requirement already satisfied: huggingface-hub<1.0,>=0.11.0 in
/opt/conda/lib/python3.10/site-packages (from transformers) (0.13.4)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/opt/conda/lib/python3.10/site-packages (from huggingface-
hub<1.0,>=0.11.0->transformers) (4.5.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/opt/conda/lib/python3.10/site-packages (from packaging>=20.0-
>transformers) (3.0.9)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.10/site-packages (from requests->transformers)
(2.1.1)
Requirement already satisfied: idna<4,>=2.5 in
/opt/conda/lib/python3.10/site-packages (from requests->transformers)
```

```
(3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/opt/conda/lib/python3.10/site-packages (from requests->transformers)
(1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.10/site-packages (from requests->transformers)
(2022.12.7)
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
```

```
from typing import Callable

from sklearn.metrics import accuracy_score,
precision_recall_fscore_support

from torch.utils.data import DataLoader, TensorDataset

from sklearn.model_selection import RepeatedStratifiedKFold,
train_test_split
from sklearn.preprocessing import LabelEncoder

import torch.nn as nn
from transformers import BertModel

def simplify_text(s):
    # очищаем текст от мусора
    s = re.sub(r"[^\w\s]", ' ', s)

    s = re.sub(r"\s+", '-', s)

    return s

dataset = Path("/kaggle/input/for-task") / "financial.json"


def read_dataset(threshold: int = 30, verbose: bool = False) ->
pd.DataFrame:

    data = pd.read_json(dataset)
    sub_data: pd.DataFrame = data["_source"].apply(pd.Series)
    data = pd.concat([data.drop(columns="_source"), sub_data], axis=1)
    if verbose:
        print(data.shape[0])

    # Осатвляем только  строки имеющие текст заявки
    data["complaint_what_happened"] =
data["complaint_what_happened"].map(lambda x: re.sub(r"XXXX|XX/XX/\
d{4}", "", x))
    filtered_data = data[data["complaint_what_happened"].str.len() !=
```

```
0]
    if verbose:
        print(filtered_data.shape[0])
    filtered_data =
filtered_data[~filtered_data["complaint_what_happened"].isnull()]
    if verbose:
        print(filtered_data.shape[0])
        print(f"Average character number in ticket body:
{filtered_data['complaint_what_happened'].str.len().mean():.2f}")

    #  ----
    null_prods = filtered_data["product"].isnull().sum()
    null_subs = filtered_data["sub_product"].isnull().sum()
    if verbose:
        print(f"Nan values in 'product': {null_prods}")
        print(f"Nan values in 'sub_product': {null_subs}")

    # ----
    if verbose:
        print("Fixing missing values ...")
    filtered_data["sub_product"] =
filtered_data["sub_product"].fillna(filtered_data["product"])

    null_prods = filtered_data["product"].isnull().sum()
    null_subs = filtered_data["sub_product"].isnull().sum()
    if verbose:
        print(f"Nan values in 'product': {null_prods}")
        print(f"Nan values in 'sub_product': {null_subs}")

    # ----
    keep = ["complaint_what_happened", "product", "sub_product"]
    filtered_data = filtered_data.loc[:, keep]
    filtered_data.columns = ["message", "label", "sub_label"]

    # удаляем дубликаты
    filtered_data = filtered_data.drop_duplicates(subset=["message"],
keep="first")
    filtered_data = filtered_data.dropna(axis=0)

    filtered_data.loc[:, "label"] =
filtered_data["label"].map(simplify_text)
    filtered_data = filtered_data.dropna(axis=0)
    filtered_data.loc[:, "sub_label"] =
filtered_data["sub_label"].map(simplify_text)
    filtered_data = filtered_data.dropna(axis=0)
    filtered_data["flattened_label"] = filtered_data["label"] + "_" +
filtered_data["sub_label"]
    filtered_data = filtered_data.reset_index(drop=True)
```

```
    # удаляем категроии с недостаточным числом записей
    c1 = filtered_data["label"].value_counts()
    filtered_data = filtered_data.replace(c1[c1 < threshold].index,
np.nan).dropna(axis=0)
    c2 = filtered_data["sub_label"].value_counts()
    filtered_data = filtered_data.replace(c2[c2 < threshold].index,
np.nan).dropna(axis=0)
    filtered_data = filtered_data.reset_index(drop=True)

    if verbose:
        print(f"Final dataset of size: {filtered_data.shape}")

print(filtered_data.columns.to_series().to_string(index=False))
        print(filtered_data.isnull().sum(axis=0))
        print(filtered_data.shape[0])
    return filtered_data

read_financial = read_dataset()
read_financial.head()
```

```
                                                message  \
0  Good morning my name is   and I appreciate it ...
1  I upgraded my   card in  and was told by the a...
2  Chase Card was reported on . However, fraudule...
3  On , while trying to book a     ticket, I came...
4  my grand son give me check for {$1600.00} i de...

                                                  label  \
0                                   Debt-collection
1                         Credit-card-or-prepaid-card
2  Credit-reporting-credit-repair-services-or-oth...
3  Credit-reporting-credit-repair-services-or-oth...
4                         Checking-or-savings-account

                                      sub_label  \
0                               Credit-card-debt
1  General-purpose-credit-card-or-charge-card
2              Other-personal-consumer-report
3                               Credit-reporting
4                               Checking-account

                                      flattened_label
0                   Debt-collection_Credit-card-debt
1  Credit-card-or-prepaid-card_General-purpose-cr...
2  Credit-reporting-credit-repair-services-or-oth...
3  Credit-reporting-credit-repair-services-or-oth...
4       Checking-or-savings-account_Checking-account
```

```
num_classes = read_financial.label.nunique()
num_classes
```

14

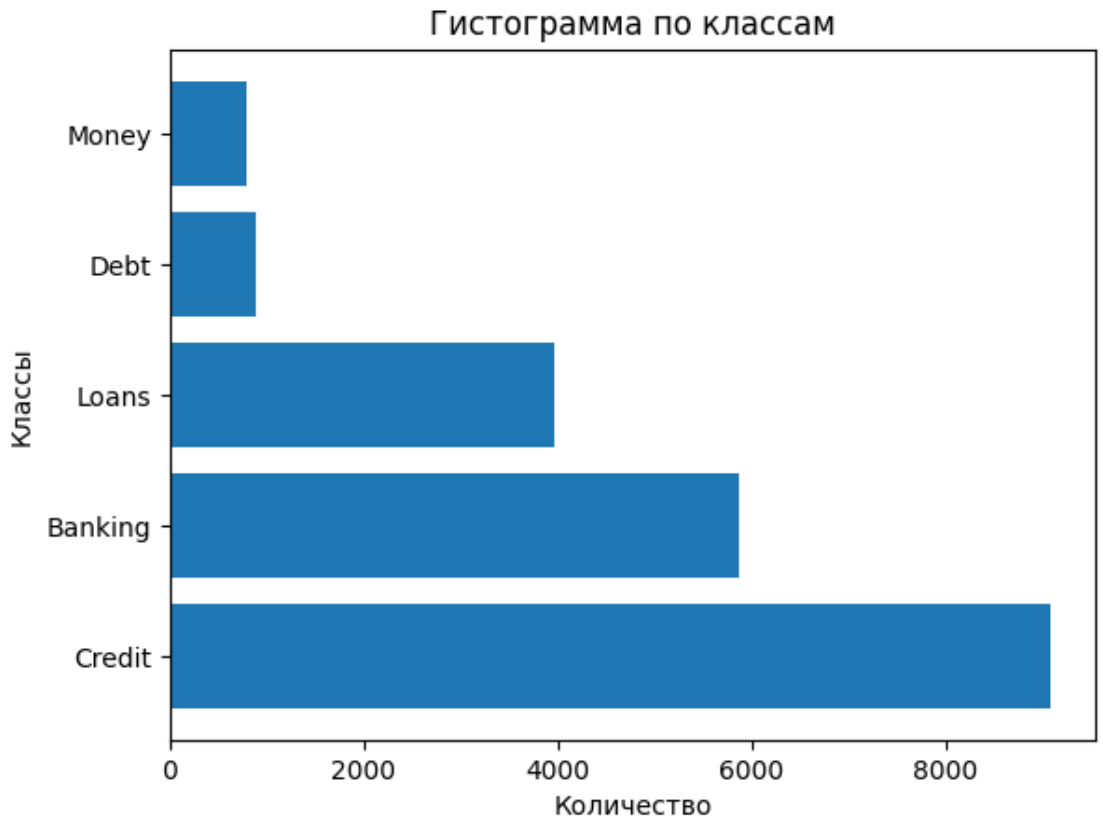
```
class_mapping = {
    'Bank-account-or-service': 'Banking',
    'Checking-or-savings-account': 'Banking',
    'Consumer-Loan': 'Loans',
    'Credit-card': 'Credit',
    'Credit-card-or-prepaid-card': 'Credit',
    'Credit-reporting': 'Credit',
    'Credit-reporting-credit-repair-services-or-other-personal-
consumer-reports': 'Credit',
    'Debt-collection': 'Debt',
    'Money-transfer-virtual-currency-or-money-service': 'Money',
    'Money-transfers': 'Money',
    'Mortgage': 'Loans',
    'Payday-loan-title-loan-or-personal-loan': 'Loans',
    'Student-loan': 'Loans',
    'Vehicle-loan-or-lease': 'Loans'
}

# Замена значений в столбце "label" с использованием словаря
соответствий
read_financial['label'] = read_financial['label'].map(class_mapping)

# Вывод измененного датафрейма
read_financial.head()
```

```
                                                 message    label  \
0  Good morning my name is   and I appreciate it ...     Debt
1  I upgraded my   card in  and was told by the a...   Credit
2  Chase Card was reported on . However, fraudule...   Credit
3  On , while trying to book a     ticket, I came...   Credit
4  my grand son give me check for {$1600.00} i de...  Banking

                                    sub_label  \
0                           Credit-card-debt
1  General-purpose-credit-card-or-charge-card
2             Other-personal-consumer-report
3                           Credit-reporting
4                           Checking-account

                                     flattened_label
0                  Debt-collection_Credit-card-debt
1  Credit-card-or-prepaid-card_General-purpose-cr...
2  Credit-reporting-credit-repair-services-or-oth...
3  Credit-reporting-credit-repair-services-or-oth...
4       Checking-or-savings-account_Checking-account
```

```
num_classes = read_financial.label.nunique()
num_classes
```

5

```
# Подсчет количества значений в каждом классе
class_counts = read_financial['label'].value_counts()

# Сортировка классов по количеству в убывающем порядке
class_counts = class_counts.sort_values(ascending=False)

# Построение упорядоченной горизонтальной гистограммы
plt.barh(class_counts.index, class_counts.values)
plt.xlabel('Количество')
plt.ylabel('Классы')
plt.title('Гистограмма по классам')
plt.show()
```




```
def regex_fun(regex: str, value: str) -> Callable:
    return lambda x: re.sub(regex, value, x)


# def remove_linux_garbage(data):
#     """
#     Taken from: https://arxiv.org/abs/1807.02892
#     Linux data contains lots of garbage, e.g. memory addresses -
0000f800
```

```
#     """

#     def is_garbage(w):
#         return len(w) >= 7 and sum(c.isdigit() for c in w) >= 2

#     data = data.map(lambda s: ' '.join(map(lambda w: w if not
is_garbage(w) else ' ', s.split())))
#     return data


def text_cleanup(msg, remove_garbage: bool = False):
    msg = msg.map(regex_fun("\r", " "))
    msg = msg.map(regex_fun("\n", " "))

    msg = msg.map(regex_fun(r"http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|
[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+", ""))

    msg = msg.map(regex_fun(r"(\w+)0x\w+", ""))

    return msg.map(str.lower)

# как выглядит очищенный текст
text_cleanup(read_financial.message)
```

```
0        good morning my name is   and i appreciate it ...
1        i upgraded my   card in  and was told by the a...
2        chase card was reported on . however, fraudule...
3        on , while trying to book a     ticket, i came...
4        my grand son give me check for {$1600.00} i de...
                               ...
20585    my husband passed away. chase bank put check o...
20586    after being a chase card customer for well ove...
20587    on wednesday, xx/xx/ i called chas, my   visa ...
20588    i am not familiar with  pay and did not unders...
20589    i have had flawless credit for 30 yrs. i've ha...
Name: message, Length: 20590, dtype: object
```

```
def split_dataset(dataframe, test_size=0.2, max_length=128):
    # Инициализация токенизатора BERT
    tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

    # Извлечение признаков и меток классов из датафрейма
    X = text_cleanup(dataframe["message"]).tolist()
    y = dataframe["label"].tolist()

    # Преобразование меток классов в числовые значения
    label_encoder = LabelEncoder()
    y = label_encoder.fit_transform(y)
    class_mapping = dict(zip(label_encoder.classes_,
label_encoder.transform(label_encoder.classes_)))
```

```
    print (class_mapping)
    # Токенизация текстовых данных
    X_tokenized = tokenizer.batch_encode_plus(
        X,
        add_special_tokens=True, # Добавление специальных токенов
[CLS] и [SEP]
        max_length=max_length, # Ограничение длины входных данных
        padding='max_length', # Добавление паддинга до максимальной
длины
        truncation=True, # Обрезание текста, если он превышает
максимальную длину
        return_attention_mask=True, # Генерация маски внимания
        return_tensors='pt' # Возвращение данных в виде PyTorch
тензоров
    )

    # Конвертация меток классов в тензор PyTorch
    y = torch.tensor(y)

    # Разбиение данных на обучающую и тестовую выборки
    X_train, X_test, y_train, y_test, attention_train, attention_test,
token_train, token_test = train_test_split(X_tokenized['input_ids'],
y, X_tokenized['attention_mask'], X_tokenized['token_type_ids'],
test_size=test_size, random_state=42)

    # Возвращение готовых для дообучения BERT данных
    return X_train, X_test, y_train, y_test, attention_train,
attention_test, token_train, token_test

X_train, X_test, y_train, y_test, attention_train, attention_test,
token_train, token_test = split_dataset(read_financial)
```


```
{"model_id":"cde4c81fb79a4ad198237b05e738ffaf","version_major":2,"vers
ion_minor":0}

{"model_id":"608da0e7ef714c30a22f975497c64d03","version_major":2,"vers
ion_minor":0}

{"model_id":"bc6ad639a4f64e4e8d97251cf913c2bd","version_major":2,"vers
ion_minor":0}
```


```
{'Banking': 0, 'Credit': 1, 'Debt': 2, 'Loans': 3, 'Money': 4}

# Создание датасетов PyTorch
train_dataset = TensorDataset(X_train, attention_train, token_train,
y_train)
test_dataset = TensorDataset(X_test, attention_test, token_test,
y_test)

# Создание DataLoader'ов для обучения и тестирования модели
batch_size = 64
```

```
train_loader = DataLoader(train_dataset, batch_size=batch_size,
shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size)

class BertClassifier(nn.Module):
    def __init__(self, num_classes):
        super(BertClassifier, self).__init__()

        self.bert1 = BertModel.from_pretrained('bert-base-uncased')
        self.bert1.requires_grad_(False)  # Замораживаем веса

        self.bert2 = BertModel.from_pretrained('bert-base-uncased')
        self.bert2.requires_grad_(False)  # Замораживаем веса

        self.fc = nn.Linear(self.bert1.config.hidden_size +
self.bert2.config.hidden_size, num_classes)
        self.dropout = nn.Dropout(0.1)

    def forward(self, input_ids, attention_mask, token_train):
        pooled_output1 = self.bert1(input_ids=input_ids,
attention_mask=attention_mask, token_type_ids=token_train)[1]
        pooled_output2 = self.bert2(input_ids=input_ids,
attention_mask=attention_mask, token_type_ids=token_train)[1]

        combined_output = torch.cat((pooled_output1, pooled_output2),
dim=1)
        combined_output = self.dropout(combined_output)

        logits = self.fc(combined_output)
        return logits

model = BertClassifier(num_classes=num_classes)
```

{"model_id":"3fc8defee6644b49b045097068abacfd","version_major":2,"vers
ion_minor":0}

```
Some weights of the model checkpoint at bert-base-uncased were not
used when initializing BertModel:
['cls.predictions.transform.LayerNorm.bias',
'cls.predictions.transform.dense.bias', 'cls.seq_relationship.bias',
'cls.predictions.transform.dense.weight',
'cls.predictions.decoder.weight',
'cls.predictions.transform.LayerNorm.weight',
'cls.seq_relationship.weight', 'cls.predictions.bias']
- This IS expected if you are initializing BertModel from the
checkpoint of a model trained on another task or with another
architecture (e.g. initializing a BertForSequenceClassification model
from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the
checkpoint of a model that you expect to be exactly identical
```

```
(initializing a BertForSequenceClassification model from a
BertForSequenceClassification model).
Some weights of the model checkpoint at bert-base-uncased were not
used when initializing BertModel:
['cls.predictions.transform.LayerNorm.bias',
'cls.predictions.transform.dense.bias', 'cls.seq_relationship.bias',
'cls.predictions.transform.dense.weight',
'cls.predictions.decoder.weight',
'cls.predictions.transform.LayerNorm.weight',
'cls.seq_relationship.weight', 'cls.predictions.bias']
- This IS expected if you are initializing BertModel from the
checkpoint of a model trained on another task or with another
architecture (e.g. initializing a BertForSequenceClassification model
from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the
checkpoint of a model that you expect to be exactly identical
(initializing a BertForSequenceClassification model from a
BertForSequenceClassification model).
```

```
# Вычисление весов классов на основе их распределения в данных
class_counts = read_financial['label'].value_counts()
total_samples = len(read_financial)
class_weights = [total_samples / (class_counts[label] *
len(class_counts)) for label in sorted(class_counts.keys())]

# Преобразование весов классов в тензор
class_weights_tensor = torch.tensor(class_weights, dtype=torch.float)
class_weights_tensor = class_weights_tensor.to(device)
# Создание экземпляра функции потерь с учетом весов классов
criterion = nn.CrossEntropyLoss(weight=class_weights_tensor)
class_weights
```

```
[0.7016527517464645,
 0.45342435586875135,
 4.621773288439955,
 1.0388496468213926,
 5.252551020408164]
```

```
# Установка вычислительного устройства для обучения модели
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print('Using device:', device)

if device.type == 'cuda':
    print(torch.cuda.get_device_name(0))
    print('Memory Usage:')
    print('Allocated:',
round(torch.cuda.memory_allocated(0)/1024**3,1), 'GB')
    print('Cached:   ', round(torch.cuda.memory_cached(0)/1024**3,1),
'GB')
```

```
model.to(device)

Using device: cuda
Tesla P100-PCIE-16GB
Memory Usage:
Allocated: 0.0 GB
Cached:    0.0 GB

/opt/conda/lib/python3.10/site-packages/torch/cuda/memory.py:416:
FutureWarning: torch.cuda.memory_cached has been renamed to
torch.cuda.memory_reserved
  warnings.warn(

BertClassifier(
  (bert1): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768,
bias=True)
              (key): Linear(in_features=768, out_features=768,
bias=True)
              (value): Linear(in_features=768, out_features=768,
bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768,
bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072,
bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
```

```
            (dense): Linear(in_features=3072, out_features=768,
bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (bert2): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768,
bias=True)
              (key): Linear(in_features=768, out_features=768,
bias=True)
              (value): Linear(in_features=768, out_features=768,
bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768,
bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072,
bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
```

```
            (dense): Linear(in_features=3072, out_features=768,
bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12,
elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (fc): Linear(in_features=1536, out_features=5, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
)

train_losses = []
test_losses = []

# device = torch.device ("cpu")
# model.to(device)

from torch.optim import AdamW
optimizer = AdamW(model.parameters(), lr=1e-4)
epochs = 5
for epoch in range(epochs):
    print("Epoch: " + str(epoch))
    # Обучение модели на тренировочном наборе
    train_loss = 0.0
    model.train()
    for batch_idx, (input_ids, attention_mask, token_train, labels) in
enumerate(train_loader):
        if batch_idx % 10 == 0:
          print("Batch: " + str(batch_idx))
        input_ids, attention_mask, token_train, labels =
input_ids.to(device), attention_mask.to(device),
token_train.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(input_ids=input_ids,
attention_mask=attention_mask, token_train=token_train)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
    train_loss /= len(train_loader)
    train_losses.append(train_loss)

    # Оценка модели на тестовом наборе
    test_loss = 0.0
```

```
    model.eval()
    with torch.no_grad():
        for batch_idx, (input_ids, attention_mask, token_test, labels)
in enumerate(test_loader):
            input_ids, attention_mask, token_test, labels =
input_ids.to(device), attention_mask.to(device),
token_test.to(device), labels.to(device)
            outputs = model(input_ids=input_ids,
attention_mask=attention_mask, token_train = token_test)
            loss = criterion(outputs, labels)
            test_loss += loss.item()
    test_loss /= len(test_loader)
    test_losses.append(test_loss)

    # Вывод информации о процессе обучения
    print('Epoch [{}/{}], Train Loss: {:.4f}, Test Loss:
{:.4f}'.format(epoch+1, epochs, train_loss, test_loss))
```

```
Epoch: 0
Batch: 0
Batch: 10
Batch: 20
Batch: 30
Batch: 40
Batch: 50
Batch: 60
Batch: 70
Batch: 80
Batch: 90
Batch: 100
Batch: 110
Batch: 120
Batch: 130
Batch: 140
Batch: 150
Batch: 160
Batch: 170
Batch: 180
Batch: 190
Batch: 200
Batch: 210
Batch: 220
Batch: 230
Batch: 240
Batch: 250
Epoch [1/12], Train Loss: 1.3174, Test Loss: 1.2763
Epoch: 1
Batch: 0
Batch: 10
Batch: 20
```

```
Batch: 30
Batch: 40
Batch: 50
Batch: 60
Batch: 70
Batch: 80
Batch: 90
Batch: 100
Batch: 110
Batch: 120
Batch: 130
Batch: 140
Batch: 150
Batch: 160
Batch: 170
Batch: 180
Batch: 190
Batch: 200
Batch: 210
Batch: 220
Batch: 230
Batch: 240
Batch: 250
Epoch [2/12], Train Loss: 1.3154, Test Loss: 1.2788
Epoch: 2
Batch: 0
Batch: 10
Batch: 20
Batch: 30
Batch: 40
Batch: 50
Batch: 60
Batch: 70
Batch: 80
Batch: 90
Batch: 100
Batch: 110
Batch: 120
Batch: 130
Batch: 140
Batch: 150
Batch: 160
Batch: 170
Batch: 180
Batch: 190
Batch: 200
Batch: 210
Batch: 220
Batch: 230
Batch: 240
```

```
Batch: 250
Epoch [3/12], Train Loss: 1.3093, Test Loss: 1.2755
Epoch: 3
Batch: 0
Batch: 10
Batch: 20
Batch: 30
Batch: 40
Batch: 50

---------------------------------------------------------------------------
-----
KeyboardInterrupt                         Traceback (most recent call
last)
Cell In[36], line 18
     16     loss.backward()
     17     optimizer.step()
---> 18     train_loss += loss.item()
     19 train_loss /= len(train_loader)
     20 train_losses.append(train_loss)

KeyboardInterrupt:
```

```python
model.dropout = nn.Dropout(0.7)
# layer1 = ['pooler.dense.', 'encoder.layer.11.', 'encoder.layer.10.',
'encoder.layer.9.']
pars1 = []
for name, param in model.bert2.named_parameters():
    pars1.append(param)

for name, param in model.bert1.named_parameters():
    pars1.append(param)

for param in pars1:
    param.requires_grad = True

from torch.optim import AdamW
optimizer = AdamW(model.parameters(), lr=1e-7)
epochs = 3
for epoch in range(epochs):
    print("Epoch: " + str(epoch))
    # Обучение модели на тренировочном наборе
    train_loss = 0.0
    model.train()
    for batch_idx, (input_ids, attention_mask, token_train, labels) in
enumerate(train_loader):
        if batch_idx % 10 == 0:
          print("Batch: " + str(batch_idx))
        input_ids, attention_mask, token_train, labels =
input_ids.to(device), attention_mask.to(device),
```

```
token_train.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(input_ids=input_ids,
attention_mask=attention_mask, token_train=token_train)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
    train_loss /= len(train_loader)
    train_losses.append(train_loss)

    # Оценка модели на тестовом наборе
    test_loss = 0.0
    model.eval()
    with torch.no_grad():
        for batch_idx, (input_ids, attention_mask, token_test, labels)
in enumerate(test_loader):
            input_ids, attention_mask, token_test, labels =
input_ids.to(device), attention_mask.to(device),
token_test.to(device), labels.to(device)
            outputs = model(input_ids=input_ids,
attention_mask=attention_mask, token_train = token_test)
            loss = criterion(outputs, labels)
            test_loss += loss.item()
    test_loss /= len(test_loader)
    test_losses.append(test_loss)

    # Вывод информации о процессе обучения
    print('Epoch [{}/{}], Train Loss: {:.4f}, Test Loss:
{:.4f}'.format(epoch+1, epochs, train_loss, test_loss))

for param in pars1:
    param.requires_grad = False
```

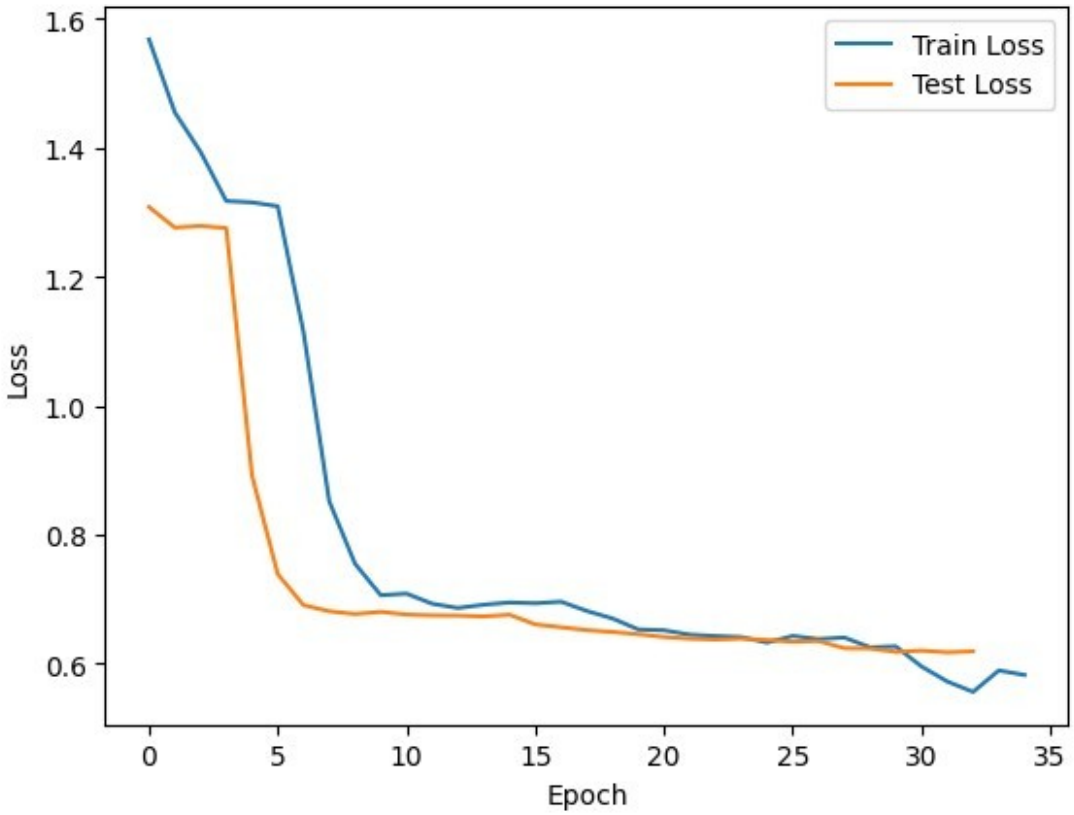
```
Epoch: 0
Batch: 0
Batch: 10
Batch: 20
Batch: 30
Batch: 40
Batch: 50
Batch: 60
Batch: 70
Batch: 80
Batch: 90
Batch: 100
Batch: 110
Batch: 120
Batch: 130
Batch: 140
```

```
Batch: 150
Batch: 160
Batch: 170
Batch: 180
Batch: 190
Batch: 200
Batch: 210
Batch: 220
Batch: 230
Batch: 240
Batch: 250
Epoch [1/3], Train Loss: 0.5890, Test Loss: 0.6172
Epoch: 1
Batch: 0
Batch: 10
Batch: 20
Batch: 30
Batch: 40
Batch: 50
Batch: 60
Batch: 70
Batch: 80
Batch: 90
Batch: 100
Batch: 110
Batch: 120
Batch: 130
Batch: 140
Batch: 150
Batch: 160
Batch: 170
Batch: 180
Batch: 190
Batch: 200
Batch: 210
Batch: 220
Batch: 230
Batch: 240
Batch: 250
Epoch [2/3], Train Loss: 0.5821, Test Loss: 0.6187
Epoch: 2
Batch: 0
Batch: 10
Batch: 20
Batch: 30

---------------------------------------------------------------------------
-----
KeyboardInterrupt                         Traceback (most recent call
last)
```

```
Cell In[49], line 28
     26 outputs = model(input_ids=input_ids,
attention_mask=attention_mask, token_train=token_train)
     27 loss = criterion(outputs, labels)
---> 28 loss.backward()
     29 optimizer.step()
     30 train_loss += loss.item()

File /opt/conda/lib/python3.10/site-packages/torch/_tensor.py:487, in
Tensor.backward(self, gradient, retain_graph, create_graph, inputs)
    477 if has_torch_function_unary(self):
    478     return handle_torch_function(
    479         Tensor.backward,
    480         (self,),
   (...)
    485         inputs=inputs,
    486     )
--> 487 torch.autograd.backward(
    488     self, gradient, retain_graph, create_graph, inputs=inputs
    489 )

File
/opt/conda/lib/python3.10/site-packages/torch/autograd/__init__.py:200
, in backward(tensors, grad_tensors, retain_graph, create_graph,
grad_variables, inputs)
    195     retain_graph = create_graph
    197 # The reason we repeat same the comment below is that
    198 # some Python versions print out the first line of a multi-
line function
    199 # calls in the traceback and some print out the last line
--> 200 Variable._execution_engine.run_backward(  # Calls into the C++
engine to run the backward pass
    201     tensors, grad_tensors_, retain_graph, create_graph,
inputs,
    202     allow_unreachable=True, accumulate_grad=True)

KeyboardInterrupt:
```

```python
for name, param in model.bert1.named_parameters():
    if param.requires_grad == False:
        print(name)
```

```python
# Построение графика изменения значения функции потерь в процессе
обучения
plt.plot(train_losses, label='Train Loss')
plt.plot(test_losses, label='Test Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




```
# Вычисление accuracy на тестовом наборе данных
model.eval()
with torch.no_grad():
    y_pred = []
    y_true = []
    for batch_idx, (input_ids, attention_mask, token_test, labels) in
enumerate(test_loader):
        input_ids, attention_mask, token_test, labels =
input_ids.to(device), attention_mask.to(device),
token_test.to(device), labels.to(device)
        outputs = model(input_ids=input_ids,
attention_mask=attention_mask, token_train=token_test)
        _, predicted = torch.max(outputs.data, 1)
        y_pred.extend(predicted.cpu().numpy())
        y_true.extend(labels.cpu().numpy())
accuracy = accuracy_score(y_true, y_pred)

# Вычисление precision, recall, f1-score на тестовом наборе данных
precision, recall, f1_score, _ =
precision_recall_fscore_support(y_true, y_pred, average='weighted')
print('Accuracy: {:.4f}, Precision: {:.4f}, Recall: {:.4f}, F1-score:
{:.4f}'.format(accuracy, precision, recall, f1_score))
```

```
Accuracy: 0.8043, Precision: 0.8423, Recall: 0.8043, F1-score: 0.8156
```

```python
# сохраняем параметры
torch.save(model.state_dict(), 'model_state_dict1.pth')

!pip install onnx
# сохраняем ONNX
model = BertClassifier(num_classes=num_classes)
model.load_state_dict(torch.load('model_state_dict1.pth'))

dummy_input_ids = torch.zeros((1, 128), dtype=torch.long)
dummy_attention_mask = torch.zeros((1, 128), dtype=torch.long)
dummy_token_id = torch.zeros((1, 128), dtype=torch.long)
torch.onnx.export(
    model,
    (dummy_input_ids, dummy_attention_mask, dummy_token_id),
    'bert_model.onnx',
    opset_version=11
)
```

```
Requirement already satisfied: onnx in /opt/conda/lib/python3.10/site-
packages (1.13.1)
Requirement already satisfied: typing-extensions>=3.6.2.1 in
/opt/conda/lib/python3.10/site-packages (from onnx) (4.5.0)
Requirement already satisfied: numpy>=1.16.6 in
/opt/conda/lib/python3.10/site-packages (from onnx) (1.23.5)
Requirement already satisfied: protobuf<4,>=3.20.2 in
/opt/conda/lib/python3.10/site-packages (from onnx) (3.20.3)
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv

Some weights of the model checkpoint at bert-base-uncased were not
used when initializing BertModel:
['cls.predictions.transform.LayerNorm.bias',
'cls.predictions.transform.dense.bias', 'cls.seq_relationship.bias',
'cls.predictions.transform.dense.weight',
'cls.predictions.decoder.weight',
'cls.predictions.transform.LayerNorm.weight',
'cls.seq_relationship.weight', 'cls.predictions.bias']
- This IS expected if you are initializing BertModel from the
checkpoint of a model trained on another task or with another
architecture (e.g. initializing a BertForSequenceClassification model
from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the
checkpoint of a model that you expect to be exactly identical
(initializing a BertForSequenceClassification model from a
BertForSequenceClassification model).
Some weights of the model checkpoint at bert-base-uncased were not
used when initializing BertModel:
['cls.predictions.transform.LayerNorm.bias',
'cls.predictions.transform.dense.bias', 'cls.seq_relationship.bias',
```

```
'cls.predictions.transform.dense.weight',
'cls.predictions.decoder.weight',
'cls.predictions.transform.LayerNorm.weight',
'cls.seq_relationship.weight', 'cls.predictions.bias']
- This IS expected if you are initializing BertModel from the
checkpoint of a model trained on another task or with another
architecture (e.g. initializing a BertForSequenceClassification model
from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the
checkpoint of a model that you expect to be exactly identical
(initializing a BertForSequenceClassification model from a
BertForSequenceClassification model).

================ Diagnostic Run torch.onnx.export version 2.0.0
================
verbose: False, log level: Level.ERROR
======================== 0 NONE 0 NOTE 0 WARNING 0 ERROR
========================
```