

# Работа с датасетом по ТВИТАМ

Проект по курсу ООП

Подготовили: Мария  
Кузовкова, Ксения  
Лазебная, Анастасия  
Тищенко





# Цель

Определить, какие из твитов датасета описывают реальные катастрофы



# План работы

**01**

**Загрузка  
данных и  
библиотек**

**04**

**Перебираем  
алгоритмы...**

Пробуем разные классификаторы  
(Logistic Regression, ансамбли,  
деревья и т.д.), токенизаторы  
(nltk/razdel)...

**02**

**Предобработка  
датасета**

**05**

**...и оцениваем  
их**

Метрики: accuracy, precision,  
recall, f-1 score, MSE

**03**

**Подготовка  
данных**

Функции токенизации, шум, train-  
test split, смотрим матрицу  
эмбедингов

**06**

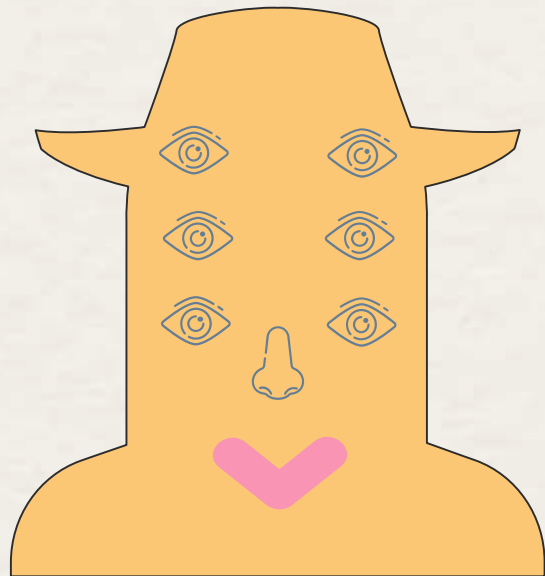
**Выбрали  
лучший  
алгоритм!**



# Загрузка данных и модулей

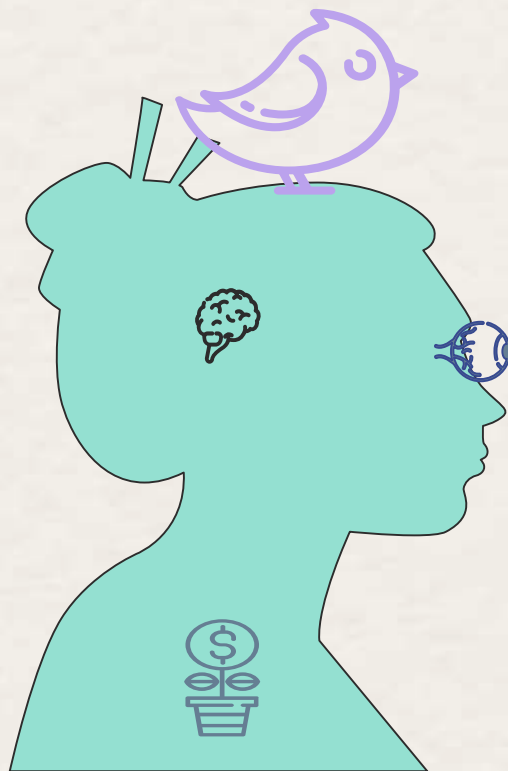
Для обучения нам нужны:

- Датасет
- Модели
- Метрики
- Лингвистические библиотеки
- TF-IDF и Count Vectorizer для получения эмбеддингов
- Инструменты регуляризации, pipeline



# Обрабатываем данные и готовимся обучать

- Удалили лишние колонки location и id
- Добавили к текстам ключевое слово
- Ввели переменную шум (стоп-слова и пунктуация)
- Делаем функции токенизации с помощью nltk и razdel
- Делаем train-test split



# Результаты тестирования разных алгоритмов

Тут пропробовали разные классификаторы. Ансамбли показали себя не очень

```
vec = TfidfVectorizer(ngram_range=(1, 1), tokenizer = word_tokenize, stop_words='english')
bow = vec.fit_transform(x_train)
clf = RandomForestClassifier(class_weight='balanced')
clf.fit(bow, y_train)
pred = clf.predict(vec.transform(x_test))
print(classification_report(pred, y_test))
```

/usr/local/lib/python3.10/dist-packages/sklearn/feature\_extraction/text.py:528: UserWarning  
warnings.warn(  
/usr/local/lib/python3.10/dist-packages/sklearn/feature\_extraction/text.py:489: UserWarning  
warnings.warn(  
precision recall f1-score support  
0 0.91 0.75 0.82 504  
1 0.63 0.85 0.72 258  
accuracy 0.77 0.88 0.78 762  
macro avg 0.77 0.80 0.77 762  
weighted avg 0.81 0.78 0.79 762

```
[ ] vec = TfidfVectorizer(ngram_range=(1, 1), tokenizer=tokenize_text)  
bow = vec.fit_transform(x_train)  
clf = xgb.XGBClassifier(random_state=0, n_jobs=1)  
clf.fit(bow, y_train)  
pred = clf.predict(vec.transform(x_test))  
print(classification_report(pred, y_test))
```

/usr/local/lib/python3.10/dist-packages/sklearn/feature\_extraction/text.py:528: UserWarning  
warnings.warn(  
precision recall f1-score support  
0 0.90 0.73 0.81 532  
1 0.57 0.81 0.67 230  
accuracy 0.76 0.76 762  
macro avg 0.74 0.77 0.74 762  
weighted avg 0.80 0.76 0.77 762



RandomForest: 63 %  
precision на тесте



XGBoost: 57 %  
precision на тесте

# Результаты тестирования разных алгоритмов

Используем униграммы, большой размер понижает качество предсказаний

```
[120] 1 vec = CountVectorizer(ngram_range=(1, 1))
      2 bow = vec.fit_transform(x_train)
      3 clf = LogisticRegression(solver='liblinear', random_state=42)
      4 clf.fit(bow, y_train)
      5 pred = clf.predict(vec.transform(x_test))
      6 print(classification_report(pred, y_test))
```

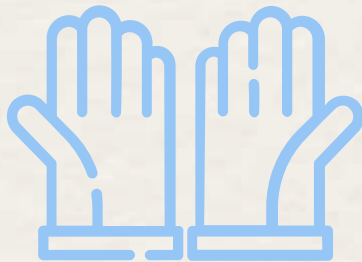
	precision	recall	f1-score	support
0	0.86	0.78	0.82	457
1	0.72	0.82	0.76	305
accuracy			0.80	762
macro avg	0.79	0.80	0.79	762
weighted avg	0.81	0.80	0.80	762

```
1 vec = CountVectorizer(ngram_range=(1, 2))
2 bow = vec.fit_transform(x_train)
3 clf = LogisticRegression(solver='liblinear', random_state=42)
4 clf.fit(bow, y_train)
5 pred = clf.predict(vec.transform(x_test))
6 print(classification_report(pred, y_test))
```

	precision	recall	f1-score	support
0	0.86	0.77	0.81	463
1	0.69	0.81	0.74	299
accuracy			0.78	762
macro avg	0.78	0.79	0.78	762
weighted avg	0.79	0.78	0.79	762



# Результаты тестирования разных алгоритмов



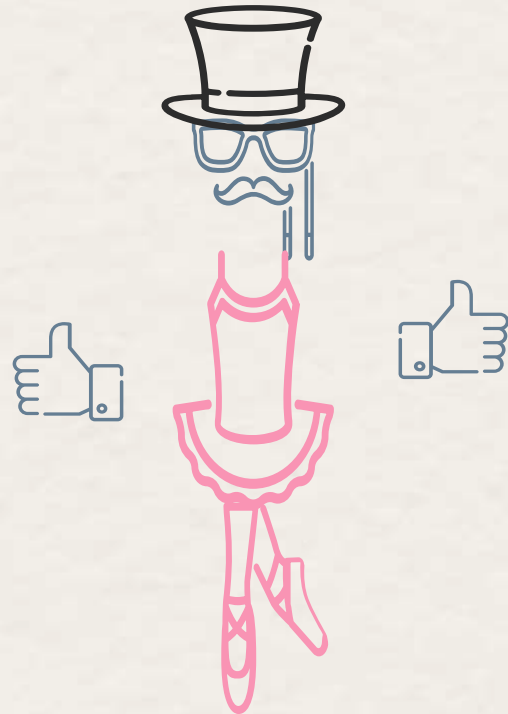
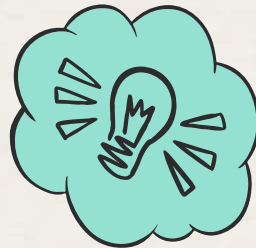
Остановились на логистической регрессии  
Также в процессе работы видно, что, когда мы  
учитываем шум, качество улучшается  
Токенизатор в итоге взят из библиотеки `nlTK`

```
vec = CountVectorizer(ngram_range=(1, 1), tokenizer = word_tokenize, stop_words=noise)
bow = vec.fit_transform(x_train)
clf = LogisticRegression(solver='liblinear', random_state=42)
clf.fit(bow, y_train)
pred = clf.predict(vec.transform(x_test))
```



**546/1178**

Наше место в соревновании на [kaggle.com](https://kaggle.com)



# Natural Language Processing with Disaster Tweets

[Submit Prediction](#)

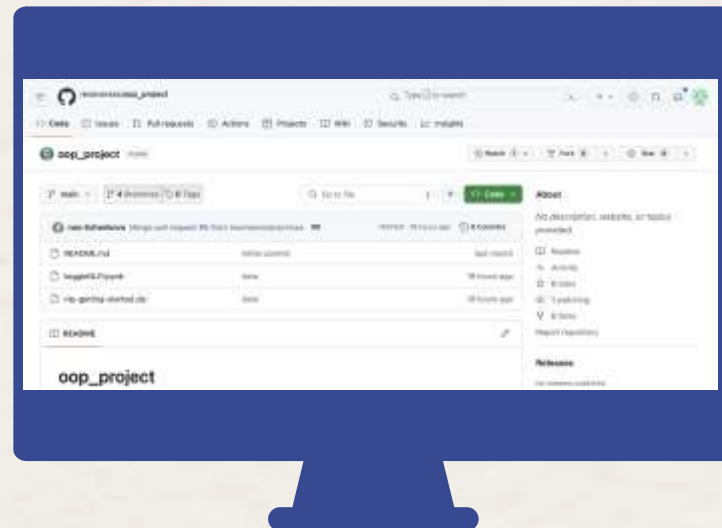
[Overview](#) [Data](#) [Code](#) [Models](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [Submissions](#)

541	parmar Komil		0.79865	3	23d
542	Soy Latte		0.79865	1	22d
543	Danilo Le Sante		0.79834	2	20d
544	winner2000		0.79834	7	8d
545	LevSaenko		0.79834	1	6d
546	<b>anastasiatish</b>		0.79834	4	1h
Your Best Entry! Your submission scored 0.79650, which is not an improvement of your previous score. Keep trying!					
547	Moongabond		0.79803	5	10d
548	Monika Kobelárová		0.79803	7	2d
549	EldorRa		0.79773	1	2mo



# Репозиторий проекта:

[https://github.com/resonansss/oop\\_project](https://github.com/resonansss/oop_project)



A brown horse with a white blaze on its face is running towards the viewer in a snowy field. The horse's mane and tail are flowing in the wind. In the background, there are snow-covered mountains and some trees. A light blue rectangular box is positioned at the bottom of the image, containing the text "Спасибо за внимание!".

**Спасибо за внимание!**