1. Категории классификатора:

`category = ['Спорт', 'Культура', 'Интернет и СМИ', 'Наука и техника', 'Экономика']`

2. Для обработки текста реализуются 3 метода:

2.1 Предобработка, которая включает в себя:

`def remove_punctuation(text)` – убирает пунктуацию;

`def remove_numbers(text)` – убирает цифры;

`def remove_multiple_spaces(text)` – убирает множественные пробелы. Используемая библиотека: `nltk.corpus`, модуль: `stopwords`.

`def remove_stop_words(text)` – убирает стоп – слова.

2.2 Стемминг текста

`def stemming_text(text)` – выделяет основу слова без его окончания. Используемая библиотека: `nltk`, модуль: `SnowballStemmer`.

2.3Лемматизация текста

`def lemmatize_text(text,mystem)`- приводит глаголы в начальную форму. Используемая библиотека: `pymystem3`, модуль: `Mystem`.

Данные методы используются как для подготовки датасета нейронной сети, так и для текста, который нужно классифицировать.

3. После подготовки датасета, разделяем его на тестовую и обучающую выборку с помощью:

`X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)`

Используемая библиотека: `Scikit-learn`, модуль: `train_test_split`

X = подготовленной текст;

y = категория текста.

70% - обучающая выборка, 30% - тестовая.

4. Для обучения модели используется библиотеку `Scikit-learn`.

В работе используется классификатор, который использует алгоритм метода опорных векторов. Основная идея метода заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Алгоритм работает в предположении, что чем больше расстояние (зазор) между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора.

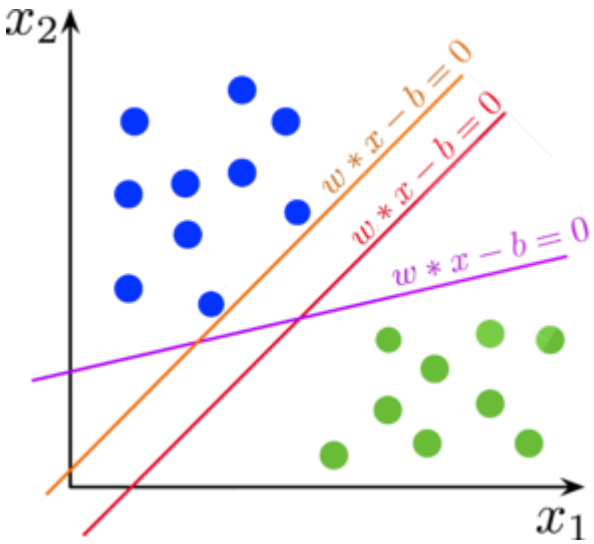

Рис.1 – Разделение двух классов объектов (blue, green)

Перед запуском обучения, реализуется конвейерная обработка, которая содержит 3 обработчика:

`CountVectorizer()` – векторизует данные в матрицу.

`TfidfTransformer()` – преобразует матрицу в нормализованное представление tf-idf.

`SGDClassifier()` – классификатор SGD реализует регуляризованные линейные модели со стохастическим градиентным спуском.

Обучение реализуется с помощью метода: `sgd.fit(X_train,y_train)`. Затем предсказываем результат классификатора на тестовой выборке: `y_pred = sgd.predict(X_test)`.

5. Точность работы классификатора

Точность: 0.9433333333333334

	precision	recall	f1-score	support
Спорт	0.94	0.87	0.90	610
Культура	0.94	0.96	0.95	591
Интернет и СМИ	0.96	0.96	0.96	584
Наука и техника	0.93	0.99	0.96	626
Экономика	0.95	0.95	0.95	589