Матчинг товаров

Команда Bananaz

Общий подход

- Предобработка текстов
- Эксперименты с моделью
- Дополнительные эвристики
- Stratify Кросс валидация

Предобработка текста

1) Лемматизация (pymorphy2):
‘Классические шкафы’ -> ‘Классический шкаф’

2) Удаление стоп слов:
‘Диван для дома’ -> ‘Диван дома’

3) Выделение чисел:
‘samsung ue50au7100ux’ -> ‘samsung ue50au7100ux 50 7100’

```
data['name'][0]
```

```
'Классическая сплит-система ROYAL CLIMA PANDORA RC-PD28HN, для комнат до 28 кв.метра,
```

```
data['prepared_name'][0]
```

```
'классический сплит-система royal clima pandora rc-pd28hn , комната 28 кв.метр , наст
```

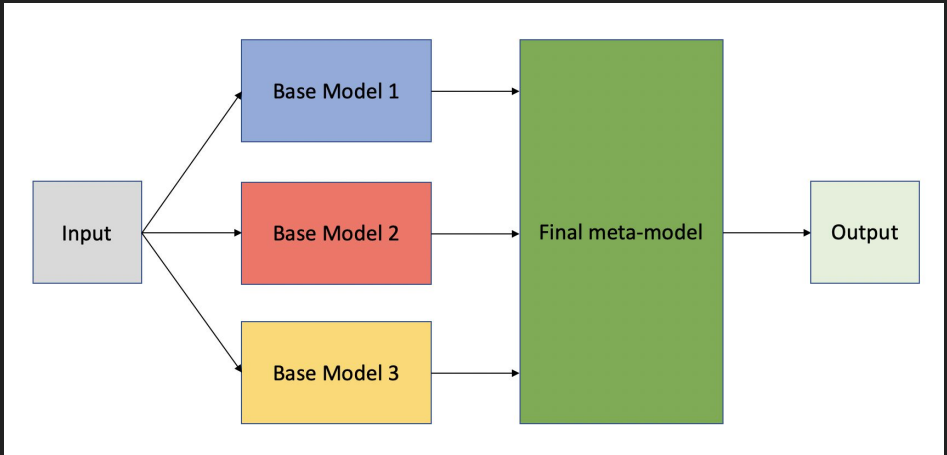
Получение эмбеддингов товаров

- **Bag of Words**
- **Bert Multilingual**
- **Tf-idf vectorize**

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Ансамбль 3-х kNN-ов



NkNN

kNN 1: Обучаем только на эмбеддингах эталонов

kNN 2: Обучаем на всех эмбеддингах X_train

kNN 3: Обучаем на усредненных эмбеддингах классов

Ансамбль: Усредняем предсказания моделей 1-3

Accuracy **0.973**

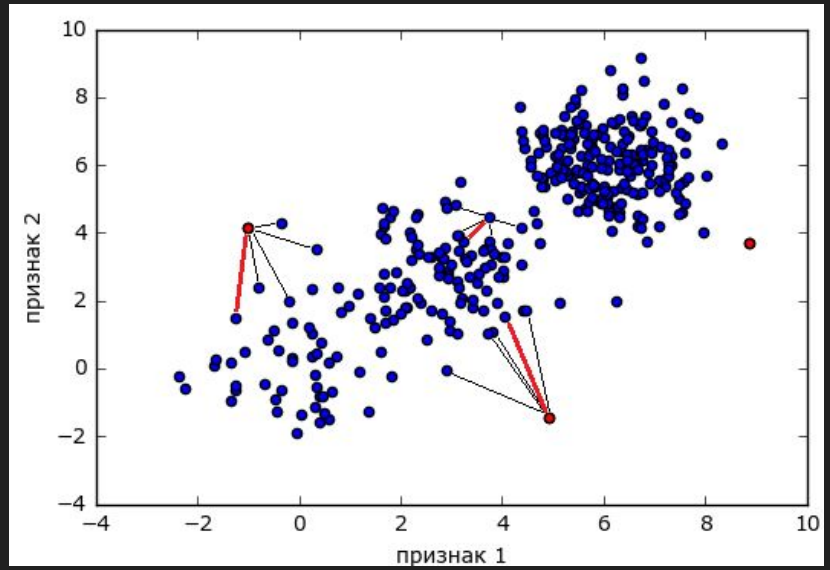
Работаем с характеристиками

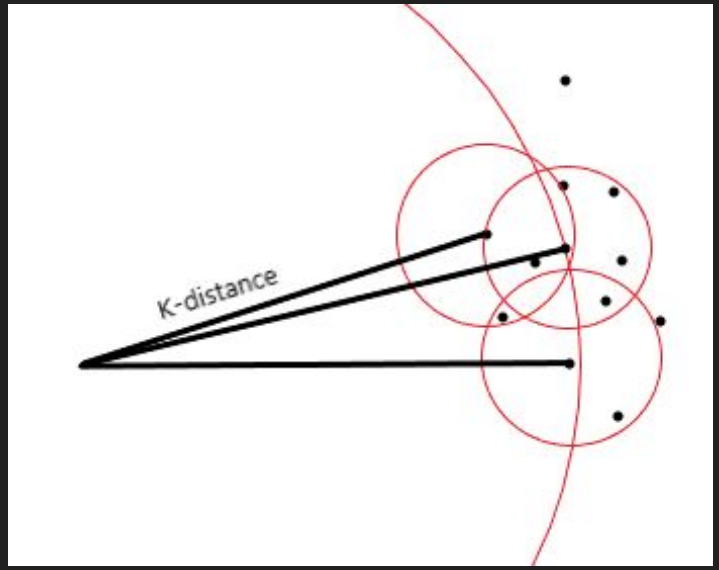
Если kNN сомневается, то давайте попробуем

- **Учитывать расстояние левенштейна**
- **Учитывать пересечения по словам**
- **Учитывать эмбеддинги описаний товаров (props)**

Accuracy **0.980**

Выявляем None-Классы





Local Outlier Factor

Финальная архитектура

Изначальное качество: kNN + tf-idf = **0.923**

kNN обученный на усредненных эмбеддингах каждого класса + tf-idf = **0.948**

pymorphy2 + удаление стоп слов = **0.953**

Выделение чисел из текста = **0.970**

Ансамбль 3 kNN’ов = **0.973**

Смотрим на описания товаров (props), если модель не уверена = **0.980**

И Local outlier factor для None классов

Итоговый accuracy на кросс валидации = **0.980**

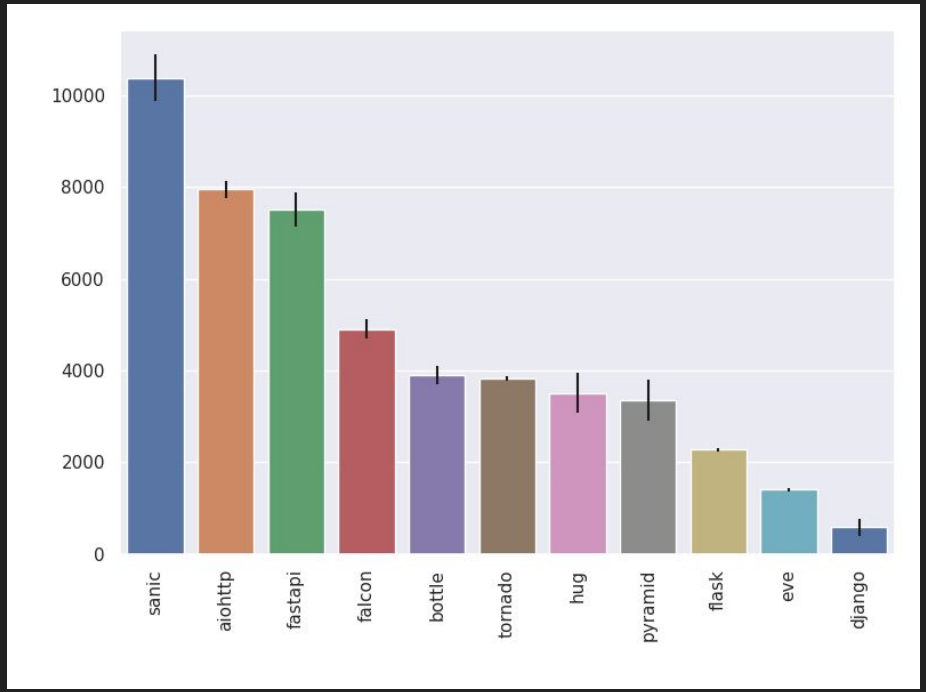
Сравнение с другими подходами

Model	Training speed	Inference speed (per 100 samples)	accuracy	Find None-class
Multilingual Bert	40h	3.5m	0.909	+
Logistic Regression	8.11s	0,0048s	0.880	+
SVC	1.72s	0.62s	0.975	-
Ансамбль KNN-ов	0.115s	0.021s	0.980	+

API на FastAPI

Почему FastAPI?

- Высокая производительность
- Легковесность
- Асинхронность
- Простота разработки



Тестирование производительности

Резюмируя

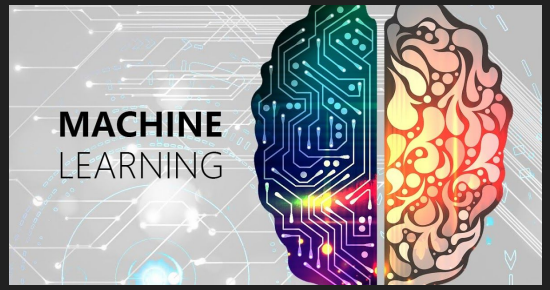
Точность модели: val - **0.980**, test - **0.925**

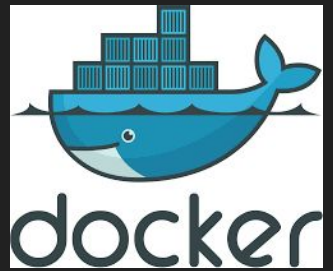
Возвращаем не один класс, а топ-5 предсказаний

Ищем None-классы

Скорость: обучение - **0.115**s, предсказания на 100 товаров - **0.02**s

Асинхронное апи в Docker





Спасибо за внимание