Студент: Малькевич Степан
Группа: CSC
Дата: 7 декабря 2016 г.

Рекомендательная система фильмов с портала Imhonet.ru
Проект по курсу «Информационный поиск»

1. Состав команды

В команде один человек: Малькевич Степан.

2. Описание системы

В связи с непопулярностью в России информационных систем, предоставляющих доступ к просмотру фильмов, а так же рекомендаций для просмотра, было решено исследовать задачу рекомендаций фильмов, на основе существующего портала для дальнейшего создания такой системы в России. Система будет предоставлять пользователю на основе его карточки (истории просмотра фильмов) новые фильмы, которые бы его заинтересовали и он бы оценил их достаточно высоко.

3. Сбор данных

В настоящей работе данные - реальная информация, полученная с портала Imhone.ru, с помощью организации РОМИП. Данные представляют из себя множество субъектов, которыми являются пользователи портала Imhonet.ru, множеством объектов - фильмы, пространство описаний транзакций - это оценки и отзывы о фильмах.

Исходные данные представлены в следующем виде:

```
<?xml version="1.0" encoding="UTF-8"?>
<table>
<columns>
<column number="0">
<name>score</name>
</column>
<column number="1">
<name>content_id</name>
</column>
<column number="2">
<name>element_id</name>
</column>
<column number="3">
<name>user_id</name>
</column>
<column number="4">
<name>text</name>
</column>
```

.

Где

- score - оценка, поставленная пользователем по 10 балльной шкале Если у отзыва стоит оценка 0, это значит, что он не оценен;
- content_id - идентификатор контента (1,2 книги, 3 фильмы);
- element_id - идентификатор книги или фильма, о котором идет речь;
- user_id - идентификатор пользователя, оставившего отзыв;
- text - текст отзыва.

4. Описание данных

Все полученные данные были преобразованы в таблицу формата tsv. На рисунке 1 представлены первые 5 строк этой таблицы. При этом в таблице количество пользователей равно 3942, количество фильмов равно 4906. Заполненность матрицы (user x item) равно 0.00081.

	score	content_id	element_id	user_id	text
0	10	3	196076	23499	Замечательный фильм, очень рекомендую.
1	8	3	218945	38132	Очень хороший фильм. Напоминает немного 6-е чу...
2	7	3	190406	38756	Фильм неплохой, если бы я не читала книгу, оце...
3	10	3	191433	38756	фильм на все времена!
4	10	3	197533	38132	Это же просто классика кино!

Рис. 1. Пример исходных данных

5. Архитектура системы

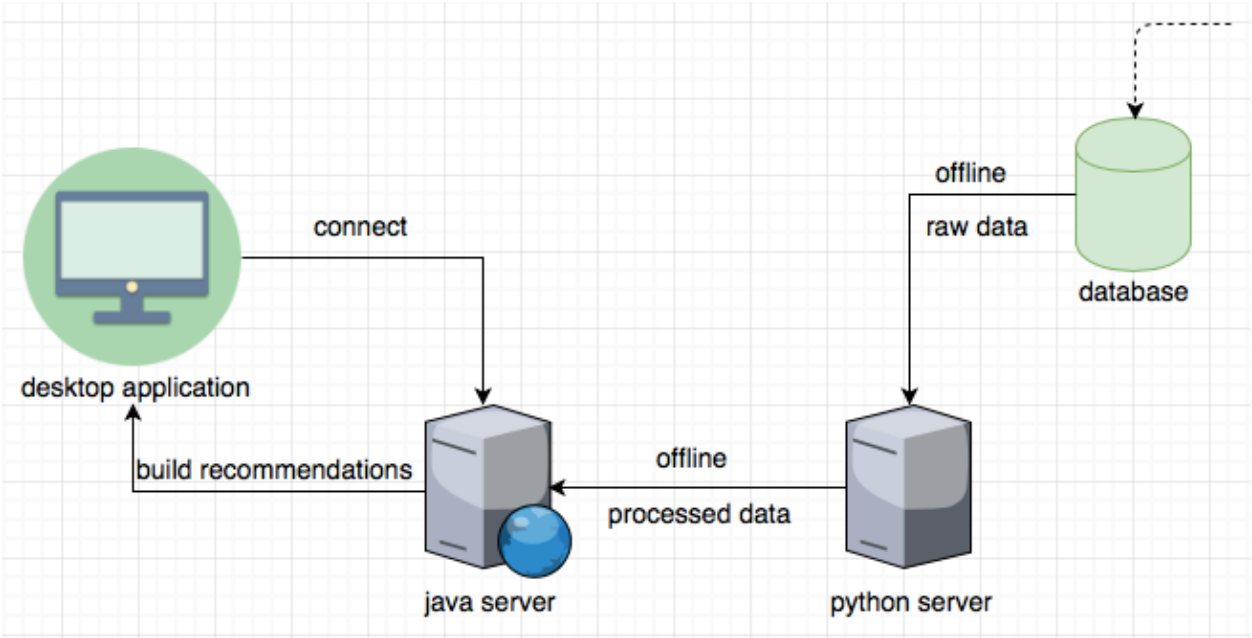

Рис. 2. Основные компоненты системы

На рисунке 2 представлена архитектура системы. Рассмотрим подробнее каждый ее компонент:

- database - хранилище данных, представлено в виде файла с данными
- python server - jupyter notebook, в котором происходит построение рекомендательных моделей
- java server - Apache Tomcat сервер, написанный с использованием библиотеки Jersey и Retrofit. Предоставляет REST API для обмены данных в формате JSON с клиентом через URL
- desktop application - оконное приложение с интерфейсом для пользователя, устанавливающее соединение с запущенным java server-ом

6. Используемые инструменты

Серверная часть системы, то есть исследование данных и их обработка, реализована на языке Python в jupyter notebook. Помимо основных библиотек для работы с данными(numpy, pandas, matplotlib, scipy) необходимо отметить следующие: gensim, pymorphy2, recsys, graphLab create.

7. Описание реализованной функциональности

Построение хороших рекомендаций - сложная задача, которую непонятно как решать. Поэтому обычно ее заменяют на построение хорошего прогноза. В дальнейшем будем говорить, что строим рекомендации, но иметь в виду под этим построение прогноза.

Для построения хорошего прогноза были применены 2 основных подхода:

(1) Colloborative filtering - используется только матрица (user x item)
(2) Conten-based recommeender systems - используется дополнительная информация о товаре и пользователе

Так же был использован подход, учитывающий explicit feedback - явное предпочтение (лайк/дизлайк) и implicit feedback - неявное предпочтение (факт покупки/просмотра).

На python server-е реализованы следующие подходы к созданию рекомендаций:

(1) Тривиальная рекомендательная система, основанная на most popular методе
(2) Построение рекомендаций на основе item-based подхода
(3) PureSVD разложение матрицы users-items
(4) Latent Factor Model
(5) Factorization Machines с дополнительными признаками

Далее следует подробное описание каждого из примененных подходов.

Most popular метод Пользователям рекомендуются фильмы в порядке убываниях их популярности - среднего рейтинга.

Item-based подход Пользователю рекомендуются фильмы, которые больше всего похожи на те, что он уже положительно оценил ранее по некоторой метрике похожести. В данной работе были попробованы 3 функции похожести: jaccard, cosine, pearson.

PureSVD Рассматривается матрица (user x item). Все пропущенные значения заменяются нулями. После чего следует обычное SVD разложение матрицы. Младшие компоненты можно убирать, что и принято делать. Обычно оставляют примерно 100 компонент. Затем можно восстановить пропущенные значения для всей исходной матрицы путем перемножения получившегося разложения.

Latent Factor Model Рассматривается матрица (user x item). Используются только заданные значения матрицы. Оптимизируется функционал, минимизирующий квадрат отклонения реального значения r_{ui} от предсказанного $\widehat{r}_{ui} = p_u^t q_i$. Физический смысл такой, что каждый пользователь/фильм представляется в виде латентного вектора, описывающего семантику пользователя/фильма. Причем эти латентные векторы бинарного формата(0 или 1). Тогда при скалярном умножении p_u на q_i получаем число, которое будет максимальным при максимальном числе совпадающих компонент. Оптимизируется обычно методом SGD, хотя часто применяют так же alternating least squares(ALS).

Factorization Machines

Это Conten-based метод, позволяющий учитывать дополнительные признаки. В данной работе были попробованы следующие наборы признаков:

- f1 - средняя оценка пользователя
- f2 - средняя оценка по фильму
- f3 - средняя длина отзыва о фильме
- f4 - средняя длина отзыва пользователя
- f5 - представление каждого отзыва в виде вектора, состоящего из частоты слов в отзыве, где словарь - это топ 10000 слов из всех отзывов train части
- f6 - выделение из train части отзывов названия фильмов, режиссеров, актеров и подобных сущностей, используя для этого Tomita Parser. Tomita позволяет извлекать из текстов факты. Извлечение фактов происходит при помощи контекстно-свободных грамматик и газеттира - словаря ключевых слов. Результирующая грамматика выглядит следующим образом:

```
FilmName → AnyWord<h−reg1, quoted>;
FilmName → AnyWord<h−reg1, l−quoted> AnyWord* AnyWord<r−quoted>;
S → FilmName;
```

Listing 1. Грамматика для Томита-парсера

В итоге, получилось вытащить 885 названий фильмов. После чего было применено, one-hot кодирование извлеченных названий фильмов.

- $f7_1$ - word2vec, основанный на всех отзывах из train части, заранее обученный на всей русской википедии(найден в сети Internet)
- $f7_2$ - doc2vec, основанный на всех отзывах из train части, обученный на отзывах из train части
- f8 - количество соседей пользователя
- f9 - средняя оценка соседей пользователя
- f10 - количество соседей фильма
- f11 - средняя оценка соседей фильма

При этом f1, f4, f5, f6, $f7_1$, $f7_2$ - признаки пользователя, f2, f3 - признаки фильма, а f8, f9 и f10, f11 - признаки от окрестности пользователя и фильма соответственно.

Для SVD разложения была использована библиотека recsys. Для всего остального библиотека GraphLab Create. Это платная библиотека, но мне удалось получить ее бесплатно на 1 год для академических целей.

8. Оценивание качества рекомендаций

Выберем некоторого пользователя u и обозначим известные для него рейтинги за R^u. В качестве тестовых рейтингов этого пользователя R^u_{test} рассмотрим 10 % известных рейтингов этого пользователя. Остальные известные рейтинги будут составлять обучающую выборку R^u_{train}. Тогда все известные рейтинги можно представить как $R^u = R^u_{train} \cup R^u_{test}$. Отсутствующие оценки обозначим за $R^u_{unknown}$. Объединив эти наборы для всех пользователей, получим наборы R_{train}, R_{test} и $R_{unknown}$.

Разобьем всю выборку на train и test случайным образом. На рисунке 3 показано полученное разбиение.

Для измерения качества рекомендаций будем использовать три метрики MSE, MAP и Spearman Rank Correlation Coefficient, описанные ниже.

8.1. Mean Squared Error.

Метрика MSE вычисляется следующим образом:

$$MSE = \frac{1}{|R_{test}|} \sum_{(u,i) \in R_{test}} (r_{ui} - \hat{r}_{ui})^2,$$

где r_{ui} — правильная оценка, а $\hat{r}_{ui}$ — оценка, предсказанная моделью.

Метрика MSE предназначена для оценки точности предсказания, ее удобно оптимизировать напрямую. Однако, нужно учесть, что MSE не лучший кандидат для оценки качества рекомендаций:

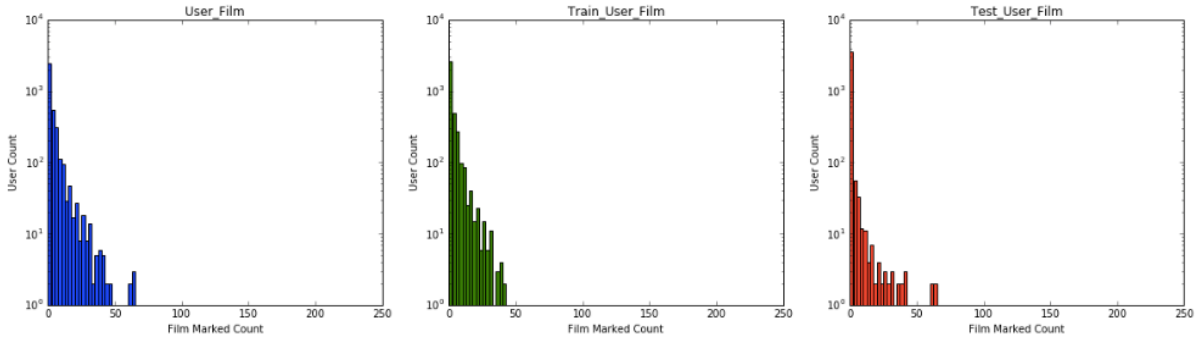

Рис. 3. Зависимость количества пользователей от количества размеченных фильмов

- MSE оценивает точность предсказания рейтингов, что в задачах рекомендаций, как правило, менее важно, нежели верное ранжирование объектов (безотносительно абсолютных значений рейтингов).
- MSE одинаково штрафует точность предсказания оценок фильмам с большим значением предпочтения (которые попадут в блок рекомендаций) и фильмам с малым значением предпочтения (длинный хвост из нерелевантных фильмов).

8.2. Mean Average Precision.

Для оценки качества рекомендаций также можно использовать метрику качества ранжирования. Для этого для каждого пользователя u предскажем оценку для всех фильмов из R^u_{test} и $R^u_{unknown}$ и отсортируем эти фильмы по убыванию предсказанного рейтинга. Ожидается, что хороший алгоритм должен выдать релевантные фильмы вверху списка. Обозначим позиции объектов в этом списке за k^u_i.

Назовем релевантными те фильмы, которые входят в R^u_{test} и имеют оценку ≥ 5. Обозначим их за Rel^u. Тогда можно считать следующую метрику качества рекомендаций для одного пользователя:

$$AP^u = \frac{1}{|Rel^u|} \sum_{(u,i) \in Rel^u} \frac{1}{k^u_i}.$$

Усреднив значение этой метрики по всем пользователями, мы получим окончательное значение метрики MAP. Пользователей без релевантных фильмов в тестовой выборке учитывать не будем.

8.3. Spearman Rank Correlation Coefficient.

Коэффициент корреляции Спирмена - мера линейной связи между случайными величинами. Корреляция Спирмена является ранговой, то есть для оценки силы связи используются не численные значения, а соответствующие им ранги. То есть, эта мера очень хорошо подходит для оценки качества ранжирования.

$$Spearman = 1 - \frac{6}{n(n-1)(n+1)} \sum_{i=1}^{n} R_i - S_i$$

, где S_i - ранг наблюдения в ряду R^u_{test}, R_i - ранг наблюдения в ряду R^u_{pred}

В результате усредним значение по всем пользователям.

9. Полученные результаты

Далее следуют полученные результаты для всех вышеописанных моделей. При этом $f7_1$ и $f7_2$ сразу плохо себя показали, так как усреднение word2vec для документа не дает истинное представление этого документа, в то время, как обучение модели doc2vec на отзывах из train части тоже не дало ничего полезного.

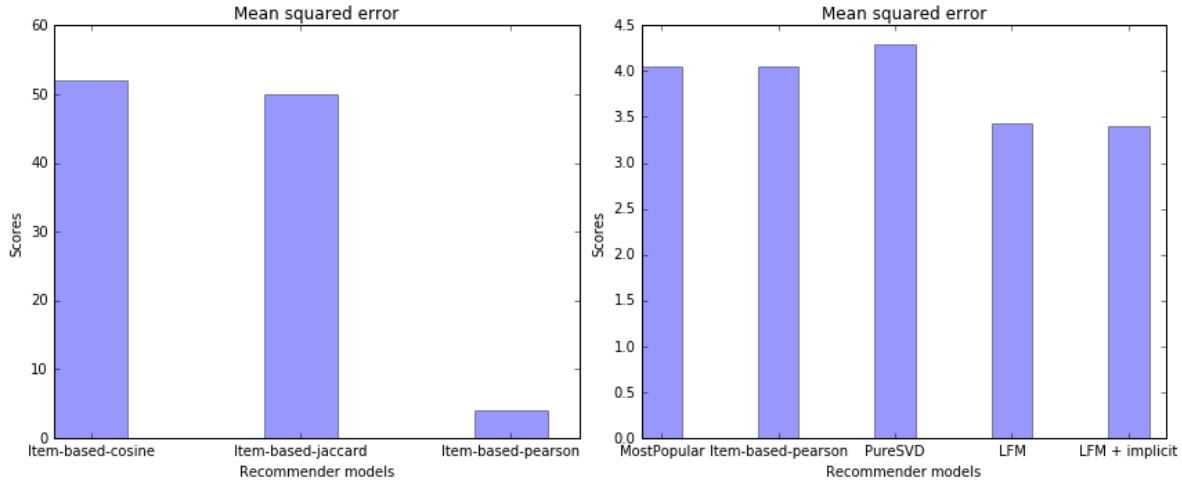

Рис. 4. Mean Squared Error

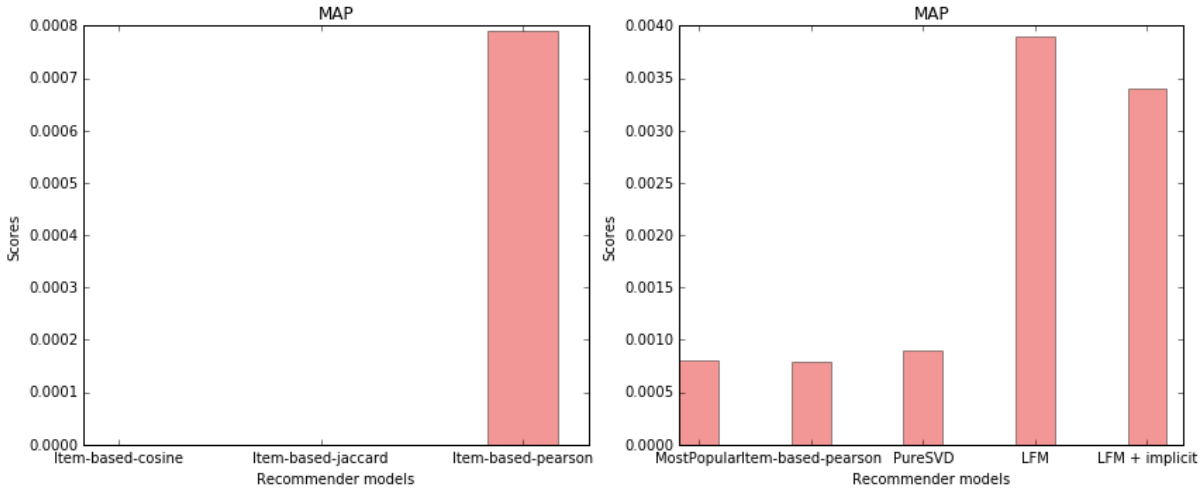

Рис. 5. Mean Average Precision

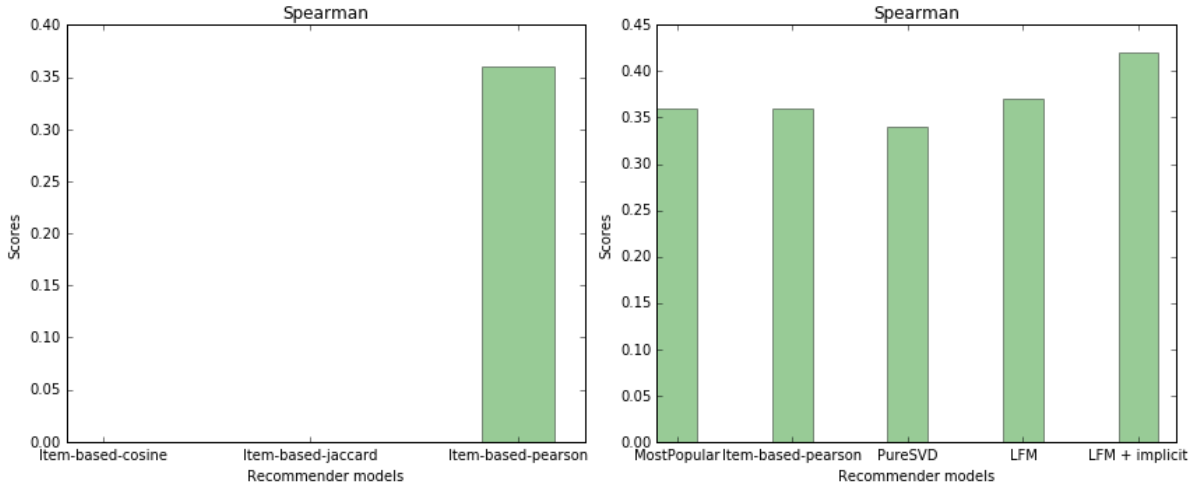

Рис. 6. Spearman Rank Correlation Coefficient

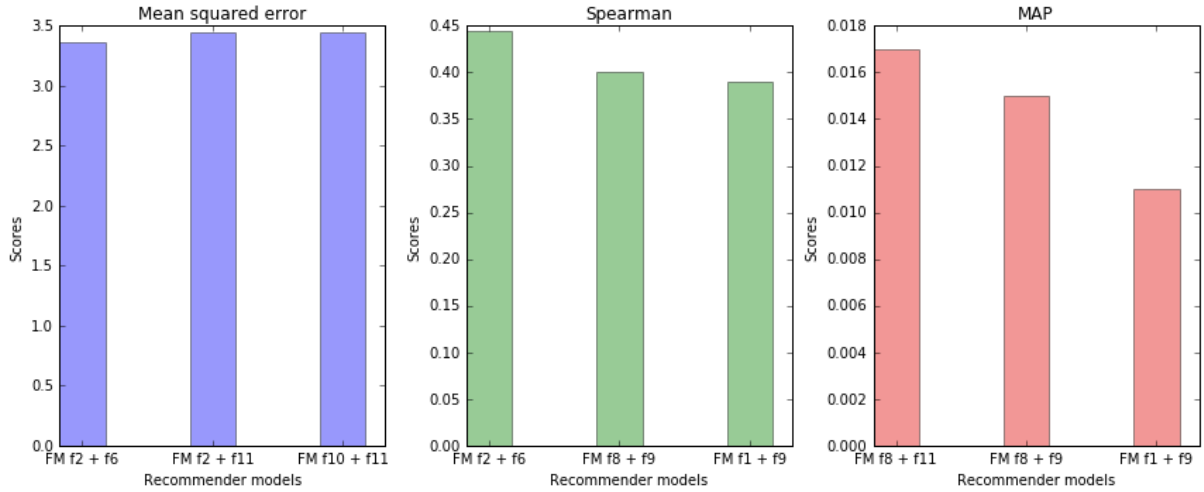

Рис. 7. Factorization Machines для попарных сочетания дополнительных признаков, топ 3

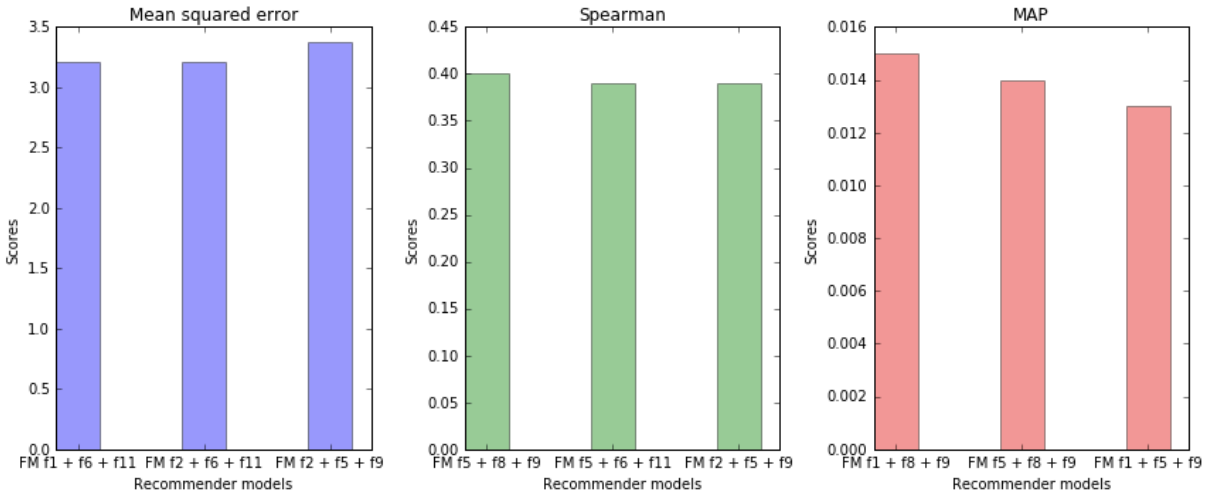

Рис. 8. Factorization Machines для сочетаний из 3 дополнительных признаков, топ 3

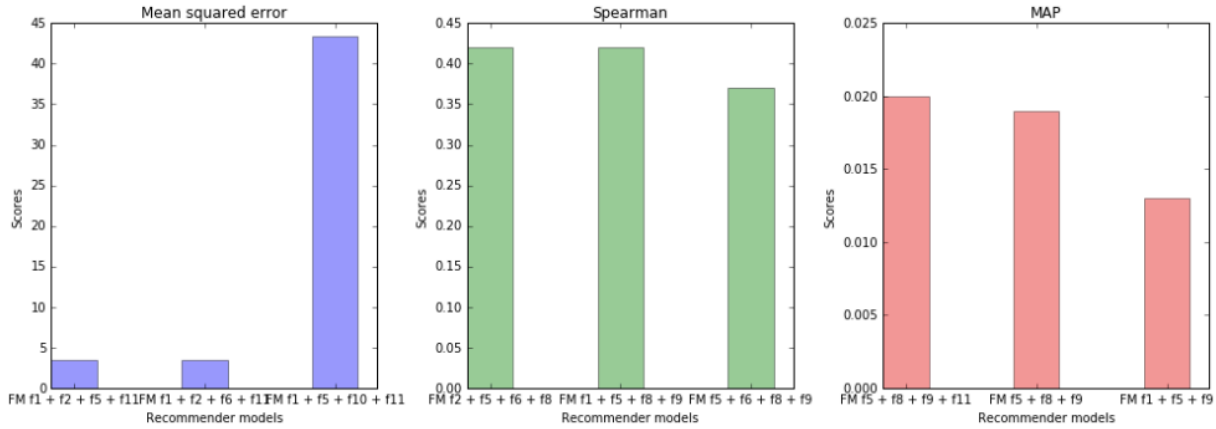

Рис. 9. Factorization Machines для сочетаний из 4 дополнительных признаков, топ 3

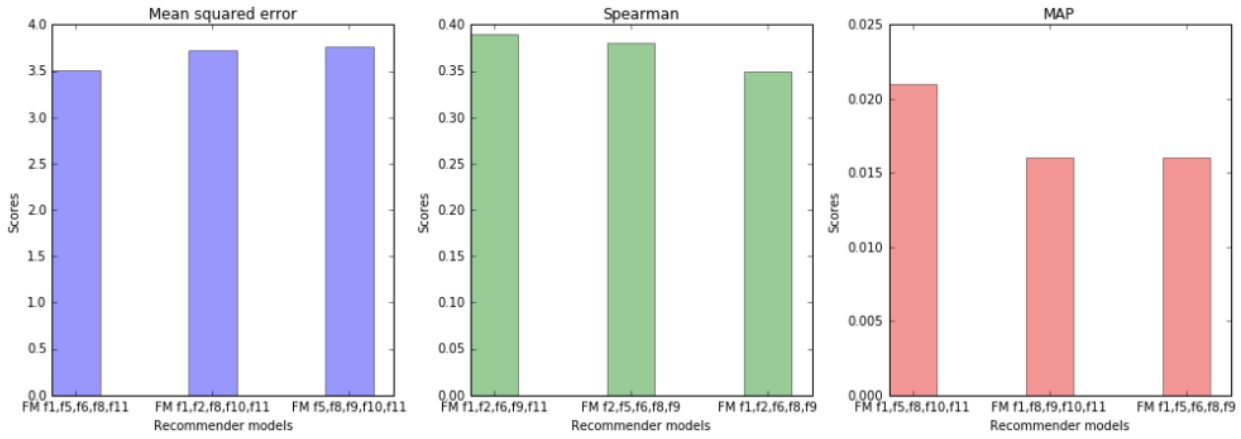

Рис. 10. Factorization Machines для сочетаний из 5 дополнительных признаков, топ 3

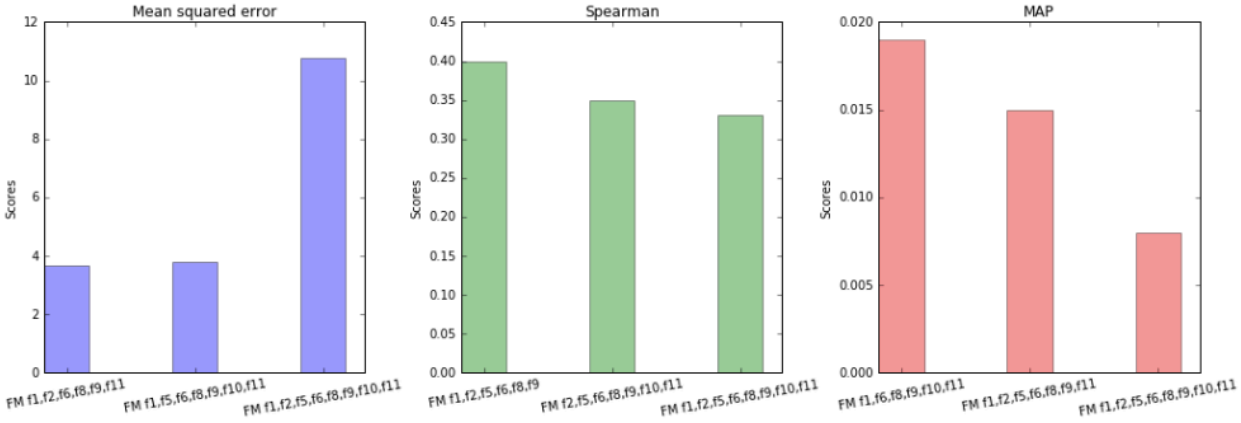

Рис. 11. Factorization Machines для сочетаний из 7,8,9 дополнительных признаков

В результате исследования было решено в продакшене использовать модель Factorization Machines с признаками f2 и f6, так как Spearman Rank Correlation Coefficient на тестовой выборке при ней максимальный из всех изученных моделей.