VK Predict CMC MSU Hackathon

Булкин Антон, Загатин Даниил

25 апреля 2025 г.

Цель задачи

- Провести бинарную классификацию пользователей: **предсказать целевую метку (target)**
- Используются два источника данных:
  - **Табличные признаки:** 1367 числовых полей
  - **Текстовые признаки:** список “токен частота” из файла `text_data.tsv`
- Целевая метрика: **ROC-AUC**
- Имеется сильный дисбаланс классов: $\sim 1.6\%$ положительного класса

Предобработка числовых признаков

1. **Downcast типов:** `float32`, `int8` — экономия памяти
2. **Импутация пропусков:** замена медианой (SimpleImputer)
3. **Нормализация распределения:** QuantileTransformer $\rightarrow$ нормальное распределение
4. **Приведение к CSR-формату:** разреженное представление для оптимизации памяти

Обработка текстовых признаков

1. Формат: “`token count token count ...`” — для каждого пользователя
2. **Парсинг словаря:** преобразование строки в `dict[token] = count`
3. **FeatureHasher:** хеширование токенов в фиксированный вектор
4. **TruncatedSVD:** понижение размерности до 50 компонент
5. **CSR-матрица:** объединение с числовыми фичами

Фрагмент кода: обработка текста

```
hasher = FeatureHasher(n_features=2**14, input_type='dict')
def gen_dicts(series):
    for s in series:
        parts = s.split()
        it = iter(parts)
        d = {}
        for tok, cnt in zip(it, it):
            d[tok] = int(cnt)
        yield d

Xh_tr = hasher.transform(gen_dicts(train_texts))
svd = TruncatedSVD(n_components=50, random_state=42)
Xsv_tr = svd.fit_transform(Xh_tr)
```

Модель и обучение

Модель: LightGBM (бустинг на решающих деревьях)

- `boosting_type='gbdt'`, `device='gpu'`
- Обработка дисбаланса: `is_unbalance=True`
- AUC как основная метрика качества

Кросс-валидация:

- Стратифицированная 5-кратная
- Ранняя остановка (`early_stopping_rounds = 30`)
- Выбор лучшего числа итераций: `best_iteration`

Фрагмент кода: обучение модели

```
params = {
    'objective': 'binary',
    'metric': 'auc',
    'boosting_type': 'gbdt',
    'device': 'gpu',
    'learning_rate': 0.01,
    'num_leaves': 128,
    'min_data_in_leaf': 30,
    'feature_fraction': 0.8,
    'bagging_fraction': 0.8,
    'bagging_freq': 1,
    'lambda_l1': 1.0,
    'lambda_l2': 1.0,
    'is_unbalance': True,
    'verbosity': 1
}
dtrain = lgb.Dataset(X_train, label=y_train)
cv = lgb.cv(params, dtrain, nfold=5, stratified=True,
            num_boost_round=2000,
            callbacks=[lgb.early_stopping(30)])
best_iter = len(cv['auc-mean'])
model = lgb.train(params, dtrain, num_boost_round=best_iter)
```

Итоговый pipeline

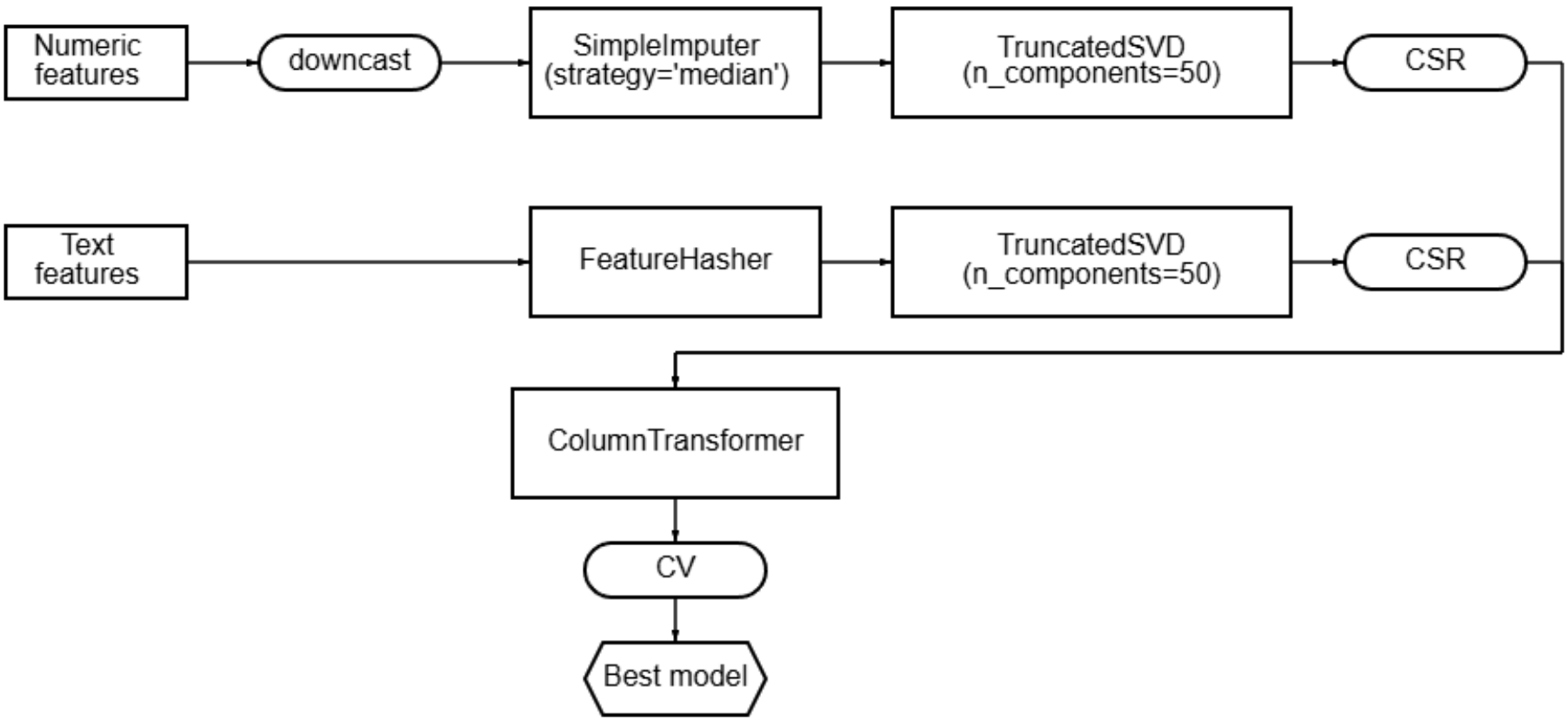

Рис.: Итоговый pipeline решения

Результаты

- **CV ROC-AUC:** 0.65+ (в зависимости от параметров)
- **Train ROC-AUC:** $\sim$0.99
- **Test ROC-AUC:** 0.66273 (по открытому leaderboard)

Спасибо за внимание!