Подход к автоматизации решений задач ML
на примере конкурса
HackerEarth Machine Learning Challenge #1

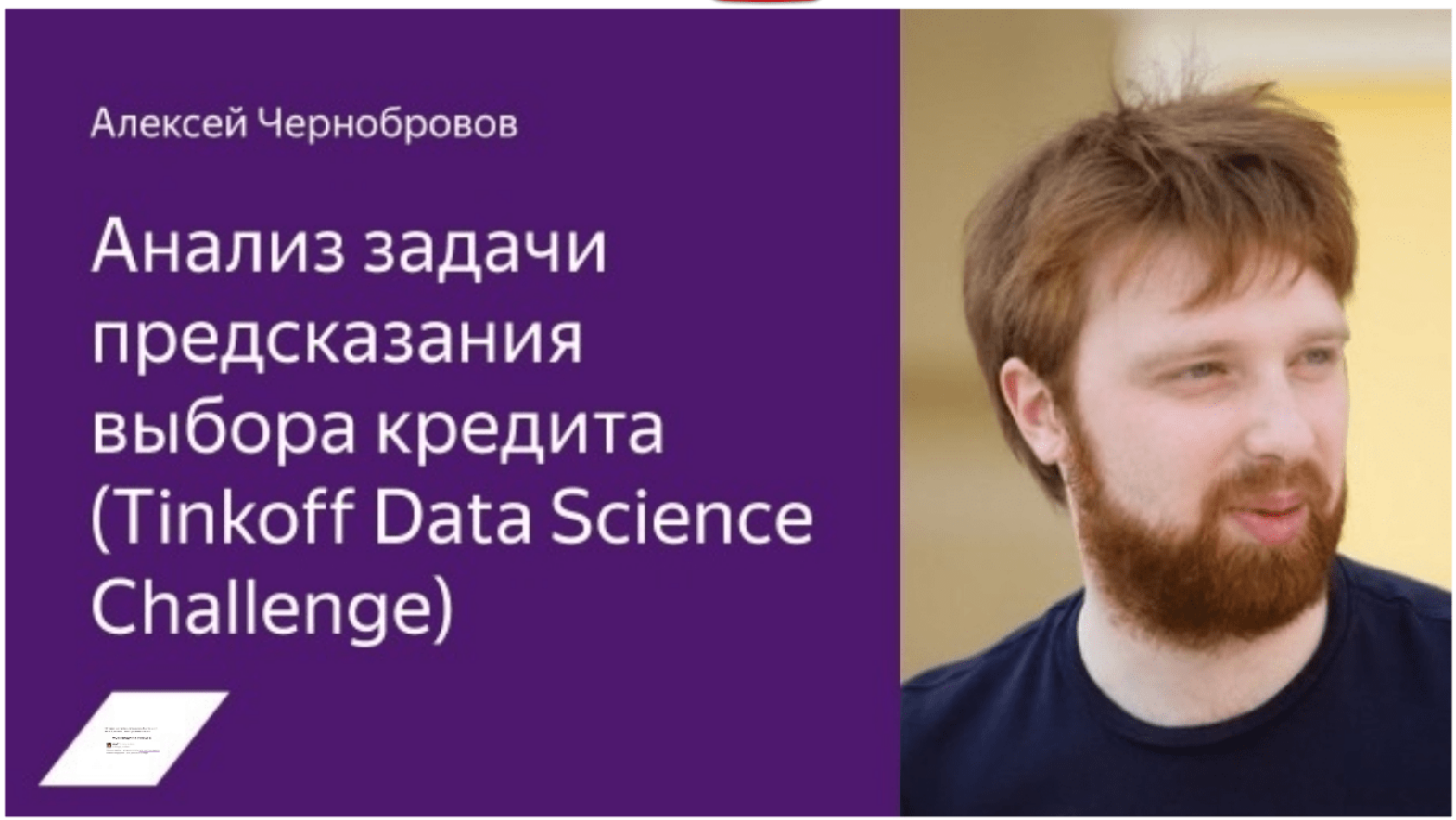
Тренировка по машинному обучению

Яндекс, 08.04.2017

Чернобровов Алексей
к.ф.-м.н.

Jet Retail

Ранее в сериале... youtu.be/1sSr6Bs8rZo



40+ часов изнурительного мозговой активности.
Это всё, конечно, очень увлекательно, но...

МОТИВАЦИЯ К РАБОТЕ

stasg7 Feb 28th at 11:26 AM
in #kaggle_crackers

Но если вкратце - в первой я опять тупо запустил скрипт, немного подправив. Сами данные не смотрел

Я подумал о том, что при решении задач:

1. Очень много повторяющихся действий

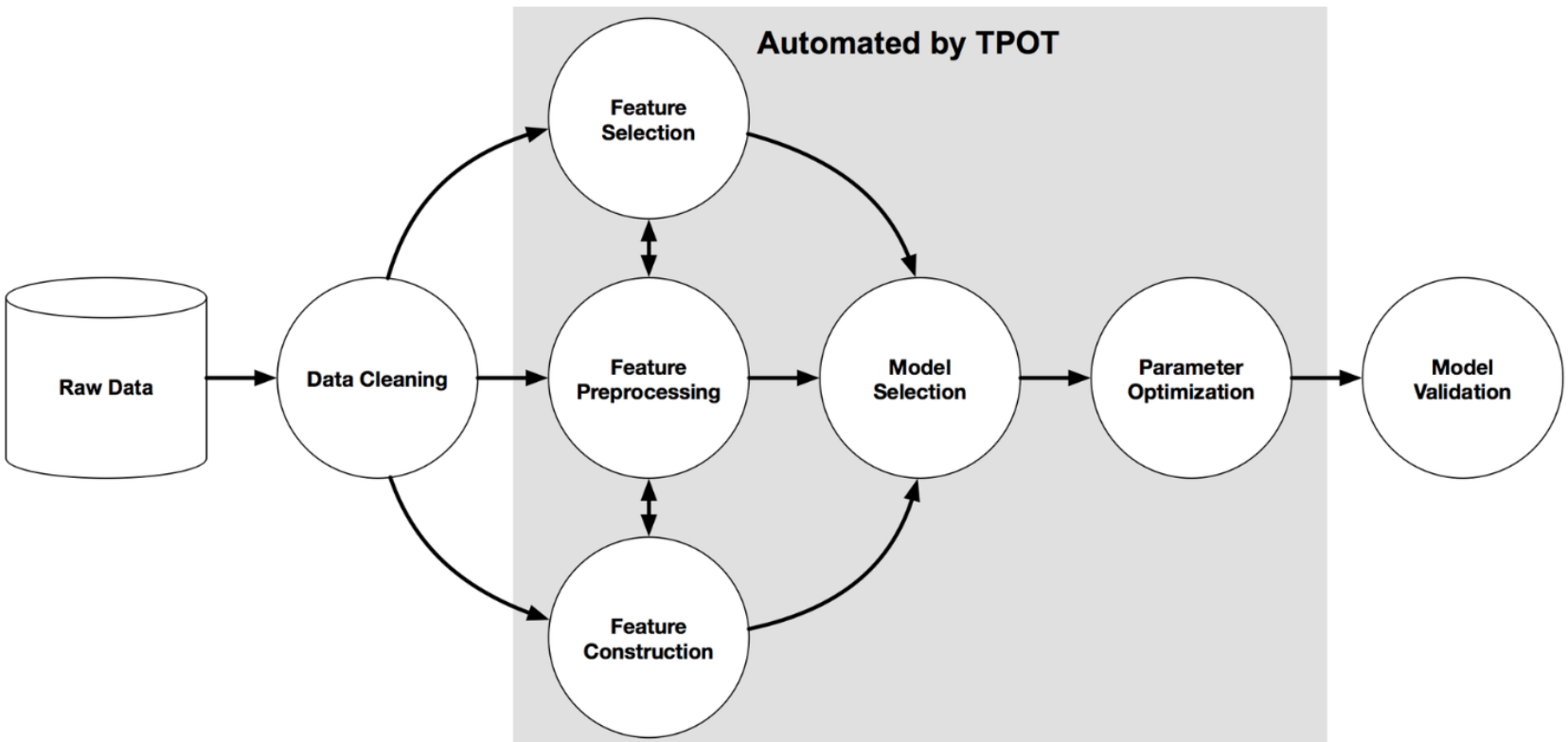
2. Очень много данных, которые можно использовать повторно

3. Надо экономить время (личное и вычислительное)

Цель: написать скрипт, который бы решал задачи по ML за меня

Конечно же, я не первый, кому пришло это в голову...

https://github.com/rhiever/tpot



Плюсы:

1. Простой интерфейс
2. Генерирует хороший код

Минусы:

1. Слабая возможность конфигурации
2. Не сохраняет промежуточные данные
3. Не стэкает
4. Часто бывает неадекватен :)

Как должен выглядеть хороший код

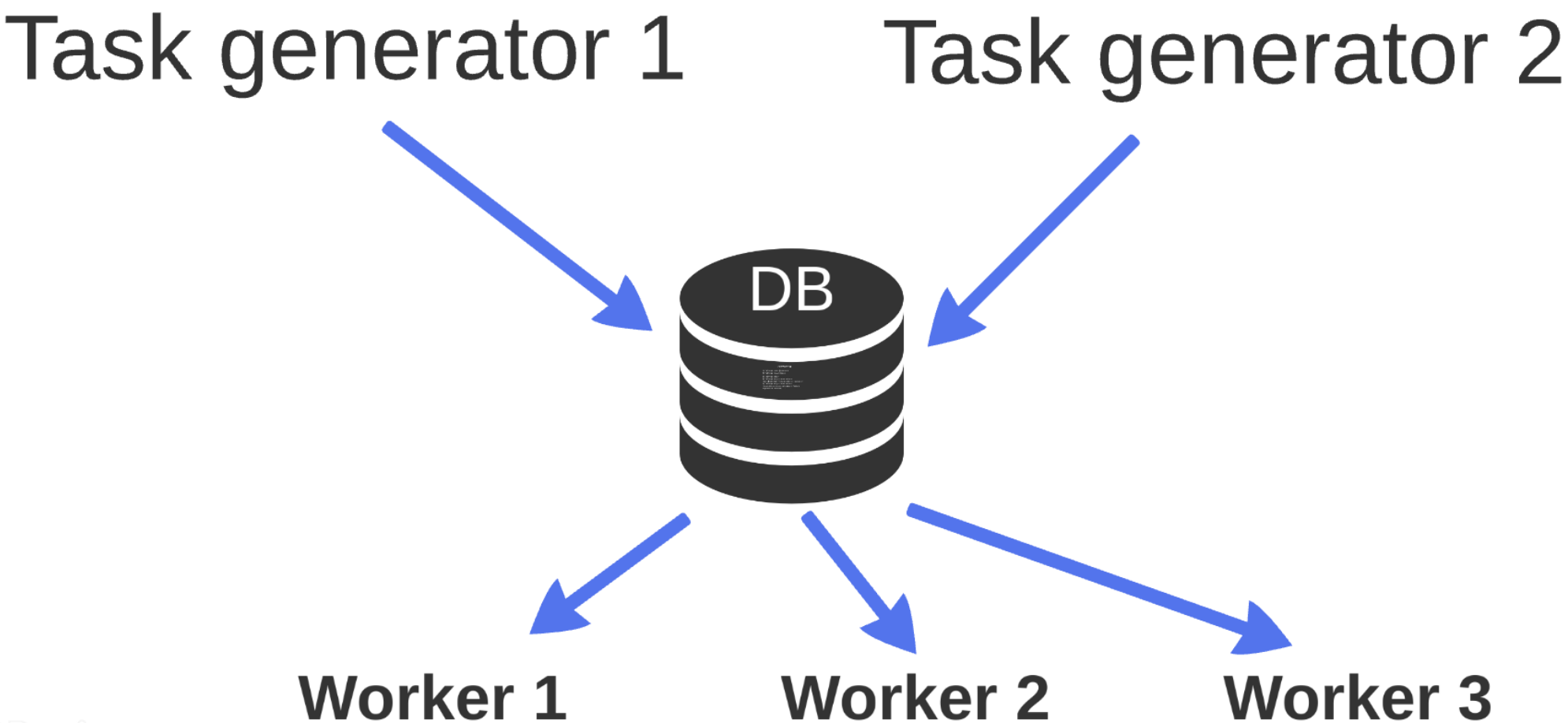
Data cleaning

```
df_train = pd.read_csv('train.csv', sep=',')
df_test  = pd.read_csv('test.csv', sep=',')
X, y, X_to_predict = DataClean(df_train,df_test)
```

Pipelines

```
pipeline1=pipeline.make_pipeline(
    DropCols([1,2,10]),
    MultiColumnHotOneEncoder([4,5,7])
    StandardScaler(),
    LogisticRegression(penalty='l2', C=2.0)
)
pipeline2=pipeline.make_pipeline(
    DropCols([4,5,7,10,11,15]),
    RobustScaler(),
    KNeighborsClassifier(n_neighbors=100)
)
#...
clf_level2 = xgb.XGBClassifier(colsample_bytree=0.7,n_estimators=330)
y_predict=Stack(clf_level2,[pipeline1,pipeline2,...]).fit(X,y).predict(X_to_predict)
```

Архитектура



Структура БД

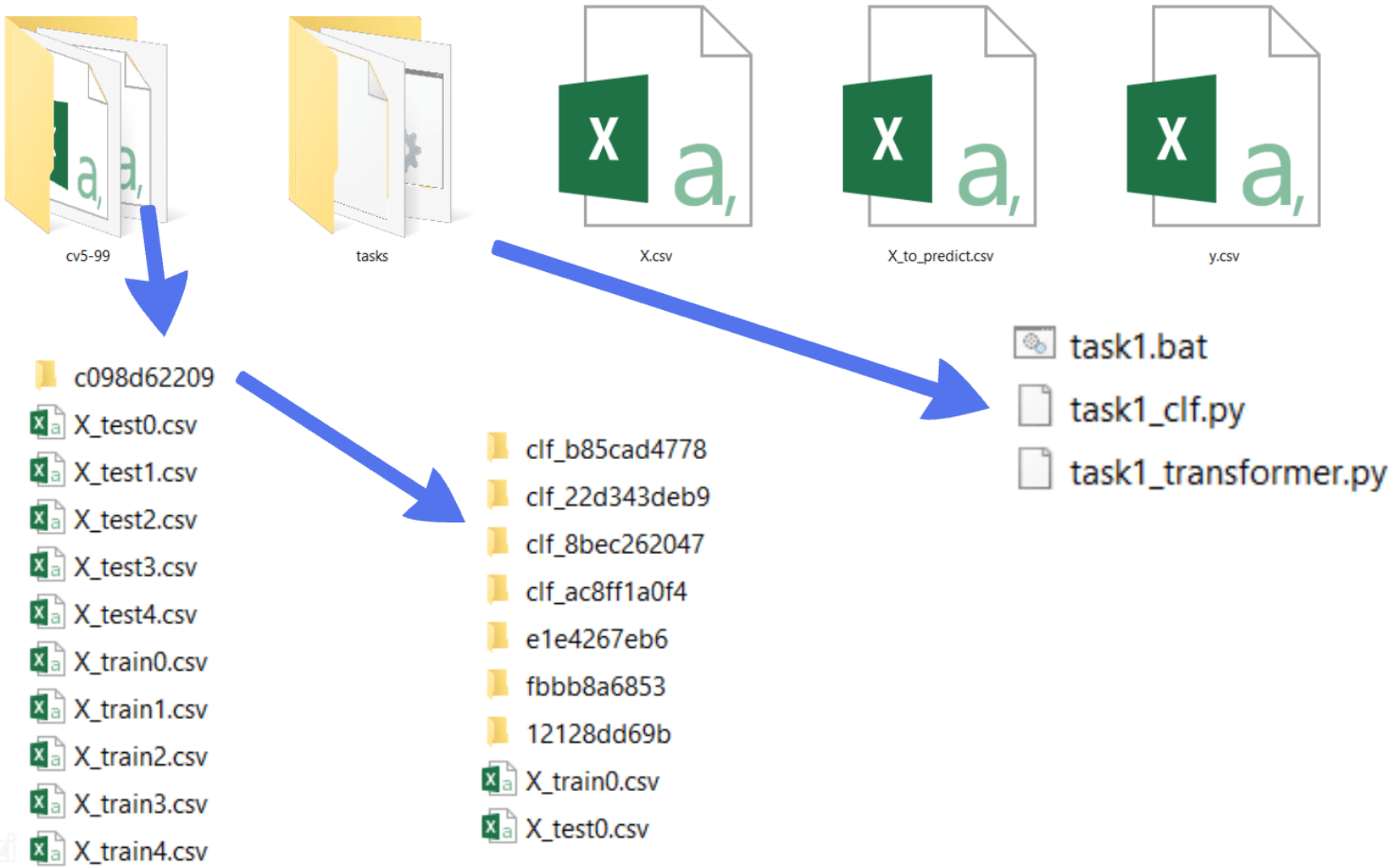
1) Таблица трансформеров
2) Таблица предикторов

3) Таблица задач
4) Таблица результатов работы трансформеров (число колонок, названия)
5) Таблица результатов работы предикторов (кросс-валидация, feature importance, intercept, ...)

Конфиг Task Generatora

- Objective
- Cross-validation (folds, random_state)
- Metrics
- Genetic algorithm (selection, mutation, ...)
- Fetures type (numeric, categorial, binary)
- Transformers
- Predictors
- *Zoo

Структура папок



В итоге получился soft:

1. Отказоустойчивый
2. Масштабируемый
3. Расширяемый
4. *Почти* без проблем "холодного старта"
5. С широкими возможностями для конфигурации
6. Стэкает из коробки :)

HackerEarth Machine Learning Challenge #1

Это весь Data cleaning

```
X = pd.read_csv('train_indessa.csv', sep=',')
X_test = pd.read_csv('test_indessa.csv', sep=',')
df = pd.concat([X, X_test])
```

```
df['term']=df['term'].apply(lambda x: int(x.replace(' months','')))
```

```
grade_dict={'A' : 1,'B' : 2,'C' : 3, 'D' : 4,'E' : 5,'F' : 6,'G' : 7}
for i in grade_dict:
    df['sub_grade']=df['sub_grade'].apply(lambda x: (x.replace(i,grade_dict[i])))
```

```
df['emp_length']=df['emp_length'].apply(lambda x: (x.replace(' years','')))
df['emp_length']=df['emp_length'].apply(lambda x: (x.replace('10+',"11")))
df['emp_length']=df['emp_length'].apply(lambda x: (x.replace('< 1 year',"0")))
df['emp_length']=df['emp_length'].apply(lambda x: (x.replace('1 year',"1")))
df['emp_length']=df['emp_length'].apply(lambda x: int(x.replace('n/a',"-1")))
```

```
cat_features0 = ['pymnt_plan', 'purpose', 'initial_list_status','application_type', 'addr_state']
for col in cat_features0:
    df[col], _ = pd.factorize(df[col])
```

```
df['zip_code1']=df['zip_code'].apply(lambda x: int(x[0]))
df['zip_code2']=df['zip_code'].apply(lambda x: int(x[0:2]))
df['zip_code']=df['zip_code'].apply(lambda x: int(x.replace('xx','')))
```

```
df['last_week_pay']=df['last_week_pay'].apply(lambda x: (x.replace('th week','')))
df['last_week_pay']=df['last_week_pay'].apply(lambda x: (x.replace('NA',"-1")))
```

```
df['desc_isnan']=pd.isnull(df['desc'])
df['emp_title_isnan']=pd.isnull(df['emp_title'])
```

```
cat_to_del=['desc','funded_amnt','grade','loan_amnt','batch_enrolled']
```

Я просто запустил свой скрипт

и...

НАЧАЛ ПРАВИТЬ БАГИ :)

Далеко не полный перечень проблем:

1. Нужно быть очень аккуратным с параметрами алгоритмов
2. Любой баг в "шаблоне" приводит к полному отсутствию предиктора
3. Нужно следить за именами колонок.
4. Делать PCA 3 раза подряд - не самая лучшая идея.
5. Такая же история с PolynomialFeatures... :)
6. У некоторых классификаторов нет predict_poba, а только predict

...

В итоге:

1. **Финальная модель:**
Стэкинг из 5 lgbm, 3 xgb, 2 adabost, 1 rf, 1 et.
Для большинства из них были разные трансфомеры.
И xgb на втором уровне.

2. **Время работы.**
Мое: 4 часа (2 часа data cleaning)
Вычислительное: примерно 60-70 часов.

3. **Результат:** 9 место из 3000+

Полезные ссылки

1. Яндекс изнутри:
как машинное обучение делает поиск умнее

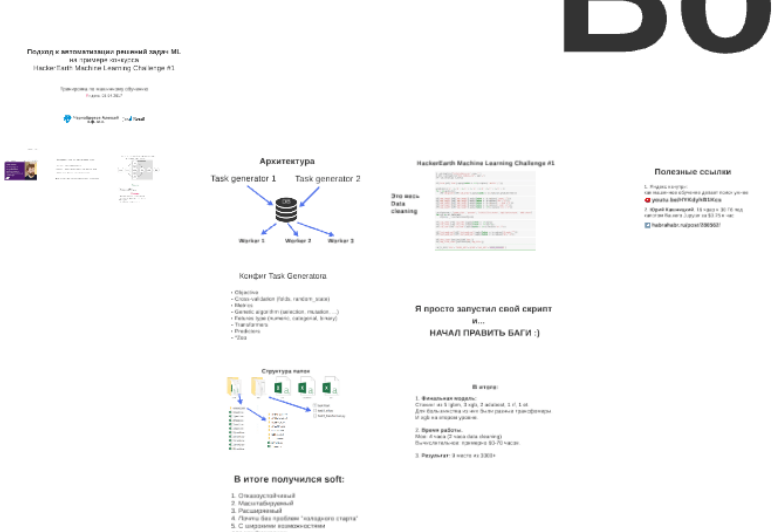
youtu.be/HYKdyhB1Kcs

2. **Юрий Кашницкий.** 16 ядер и 30 Гб под капотом Вашего Jupyter за $0.25 в час

habrahabr.ru/post/280562/

Спасибо за внимание!

Вопросы?



vk.com/chernobrovov

OpenDataScience @alex4er

Алексей Чернобровов