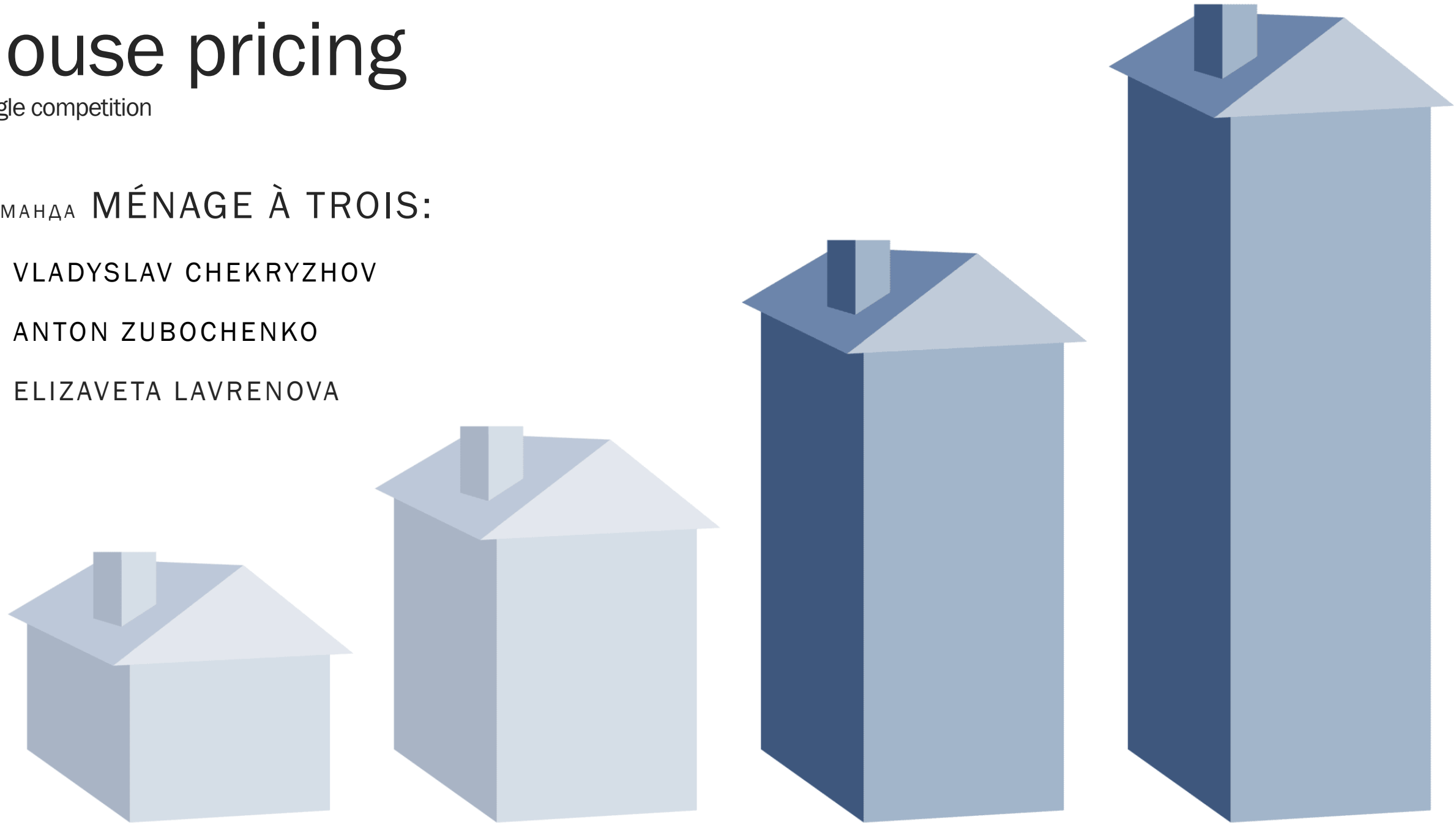
House pricing

Kaggle competition

КОМАНДА MÉNAGE À TROIS:

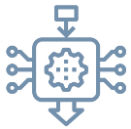
- VLADYSLAV CHEKRYZHOV
- ANTON ZUBOCHENKO
- ELIZAVETA LAVRENOVA



Задача: спрогнозировать цену дома на основе описывающих его переменных (задача регрессии).

План действий:

EDA – исследовательский анализ исходных данных, преобразование тестовой и обучающей выборок к формату, необходимому для использования в моделях.



Models – применение выбранных алгоритмов для построения регрессии, выбор модели для дальнейшего использования на основе сравнения результатов точности.

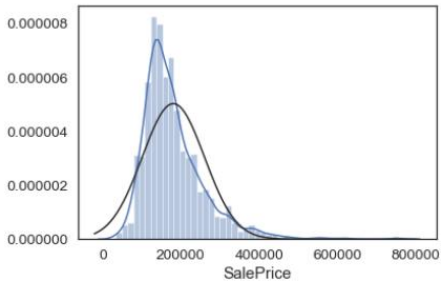
Tuning – подбор оптимальных параметров для достижения наилучшего результата при помощи использования выбранной соло-модели.

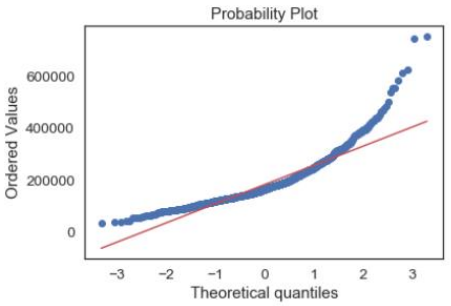
Stacking / Blending – обучение мета-алгоритмов на основе базовых моделей / комбинация выходов моделей для получения финального результата.

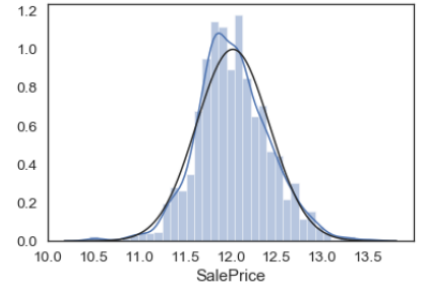
Neural network – применение полносвязной нейронной сети для решения задачи.

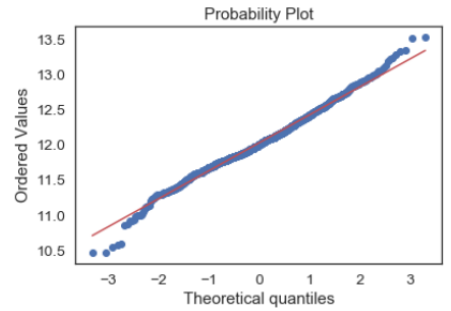
Исследование целевой переменной

- исходное распределение переменной SalePrice отличалось от симметричного нормального распределения;
- при применении операции логарифмирования к исходной переменной было получено новое значение целевой переменной.



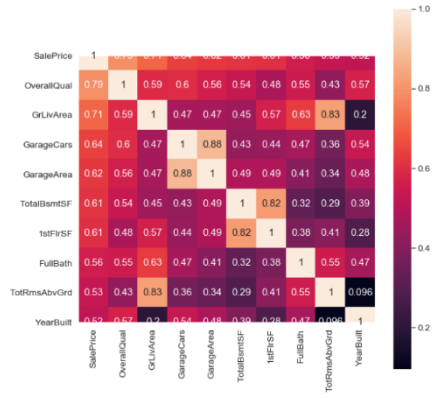


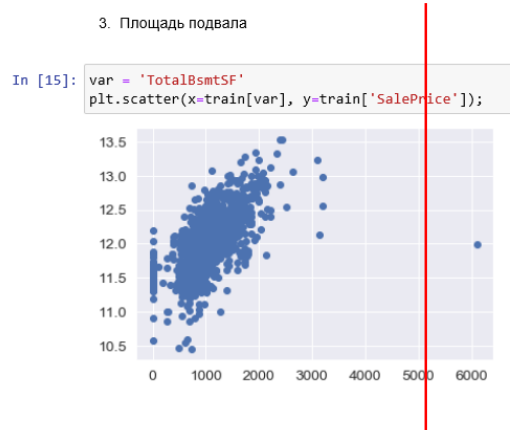




Работа с выбросами

- Для 10 наиболее коррелированных с целевой переменной признаков был произведен отбор выбросов. Они были определены на основе визуального анализа графиков и удалены из выборки.





Работа с пропусками

- Переменные с одним пропуском:
  - Числовые переменные заполнены значением моды;
  - Категориальные переменные – введено новое значение «Typ».

- Переменные с несколькими пропусками:
  - Числовые – заполнены нулями или средними значениями по фактору;
  - Категориальные – введено новое значение «None».

```
In [28]: all_features.isnull().sum().sort_values(ascending=False).head(5)

Out[28]: PoolQC          2905
         MiscFeature     2810
         Alley           2716
         Fence           2343
         FireplaceQu     1419
         dtype: int64
```

```
In [34]: all_features.isnull().sum().sort_values(ascending=False).head(5)

Out[34]: SaleCondition    0
         Foundation       0
         RoofMatl         0
         Exterior1st      0
         Exterior2nd      0
         dtype: int64
```

Преобразование бокса-кокса

- Для переменных, имеющих несимметричное распределение, было применено преобразование бокса-кокса.
- После применения преобразования из 27 факторов с несимметричным распределением осталось только 17 переменных.

```
There are 27 numerical features with Skew > 0.5 :
Mean skewnees: 4.04656877286393
```

```
There are 17 skewed numerical features after Box Cox transform
Mean skewnees: 3.5003659362844797
```

Feature engineering (добавление новых факторов)

- Добавлены переменные, являющиеся комбинациями имеющихся факторов:
  - TotalSF – суммарная площадь жилья,
  - Total_Bathrooms – суммарное количество ванных комнат
  и другие.

- Добавлены индикаторы на основе числовых значений выбранных переменных:
  - haspool – наличие бассейна,
  и другие.

Кодирование категориальных переменных

- Для кодирования категориальных переменных применена функция *pandas.get_dummies()*.

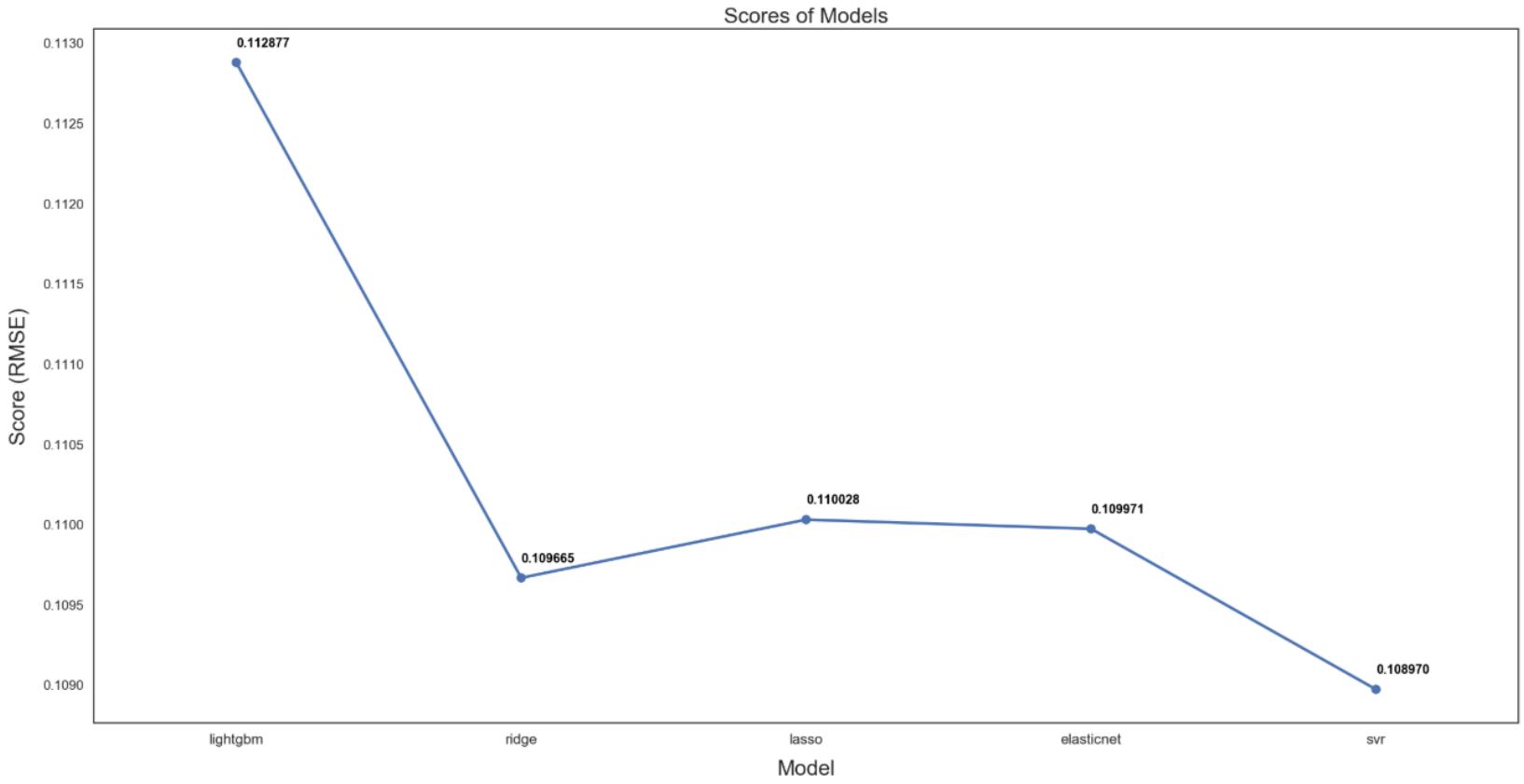
```
[30] all_features.shape

     (2914, 81)

[29] all_features_dummy = pd.get_dummies(all_features).reset_index(drop=True)
     all_features_dummy.shape

     (2914, 334)
```

Примененные модели	Средняя точность	Стандартное отклонение
LGBMRegressor	0.1129	0.0173
RidgeCV	0.1097	0.0150
LassoCV	0.1100	0.0160
ElasticNetCV	0.1100	0.0160
SVR	0.1090	0.0157

Последний результат по точности принадлежит алгоритму бустинга *lightgbm*, попробуем подобрать для него оптимальные параметры, чтобы повысить точность.

Для оценки точности (RMSE) использовалось разбиение на 16 фолдов с перемешиванием;

Для каждой модели на графике выведена средняя точность.



Модель градиентного бустинга показала худший результат на этапе отбора, подбор оптимальных параметров позволяет улучшить точность предсказания цен с помощью выбранного алгоритма.

submission.csv 7 hours ago by vladyslav chekryzhov add submission details	0.12265
submission.csv 8 hours ago by vladyslav chekryzhov add submission details	0.12166
submission.csv 8 hours ago by vladyslav chekryzhov add submission details	0.12324
submission.csv 10 hours ago by vladyslav chekryzhov add submission details	0.40302

Старая модель

In [53]:
```
# LightGBM
old_lightgbm = LGBMRegressor(objective='regression',
                             num_leaves=4,
                             learning_rate=0.01,
                             n_estimators=9000,
                             max_bin=200,
                             bagging_fraction=0.75,
                             bagging_freq=5,
                             bagging_seed=7,
                             feature_fraction=0.2,
                             feature_fraction_seed=7,
                             min_sum_hessian_in_leaf = 11,
                             verbose=-1,
                             random_state=42)

scores = {}
score = cv_rmse(old_lightgbm)
print("old lightgbm: {:.4f} ({:.4f})".format(score.mean(), score.std()))
scores['old lightgbm'] = (score.mean(), score.std())
```

```
old lightgbm: 0.1130 (0.0172)
```

Новая модель с оптимальными параметрами

In [56]:
```
light_gbm = LGBMRegressor(objective='regression',
                          num_leaves=4,
                          learning_rate=0.01,
                          n_estimators=4500,
                          max_bin=255,
                          bagging_fraction=0.65,
                          bagging_freq=4,
                          bagging_seed=2,
                          feature_fraction=0.2,
                          min_sum_hessian_in_leaf = 1,
                          verbose=-1,
                          random_state=500)
```

In [57]:
```
score = cv_rmse(light_gbm)
print("lightgbm: {:.4f} ({:.4f})".format(score.mean(), score.std()))
scores['lightgbm'] = (score.mean(), score.std())

lgb_model = light_gbm.fit(X, y_train)
submission=lgb_model.predict(X_test)
```

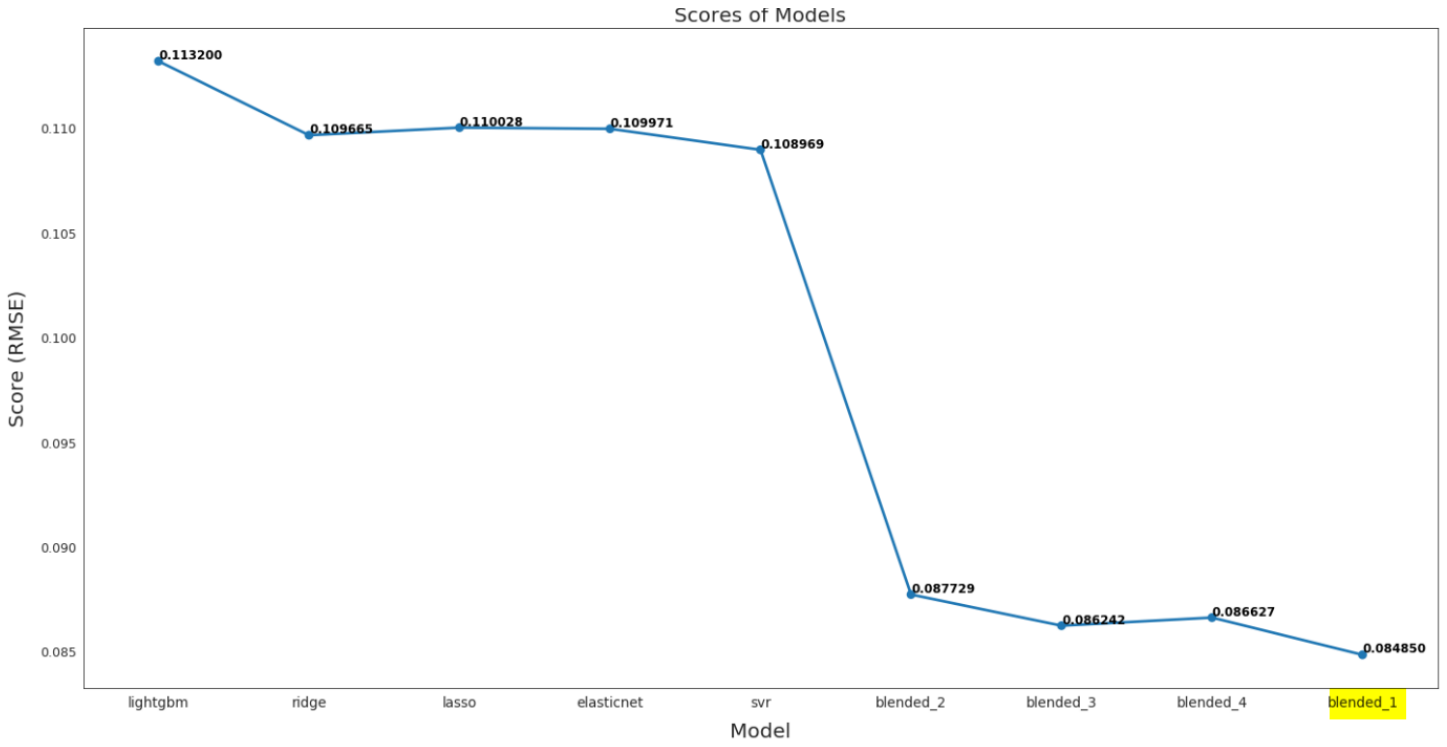
```
lightgbm: 0.1122 (0.0172)
```

Для стакинга использовался алгоритм StackingCVRegressor из пакета mlxtend.regressor. Алгоритм прогонялся на основе результатов моделей, использованных на этапе baseline.

```
----START Fit---- 2020-02-26 19:18:34.347014
Elasticnet
Lasso
Ridge
lightgbm
svr
stack_gen
```

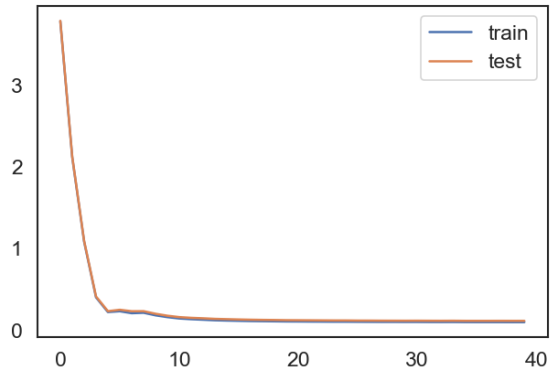
```
[ ]  score = cv_rmse(stack_gen_model)
     print("stack_gen_model: {:.4f} ({:.4f})".format(score.mean(), score.std()))
     scores['stack_gen_model'] = (score.mean(), score.std())
```

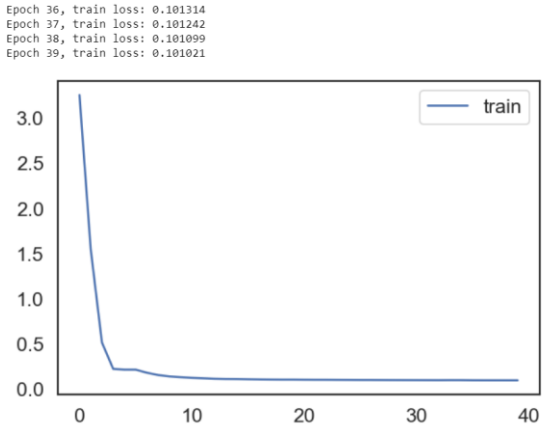
```
stack_gen_model: 0.1094 (0.0135)
```



Для блендинга были испробованы разные комбинации, лучший результат был достигнут при использовании 5 baseline – моделей без учета стакинга.

Для реализации нейросети выбран пакет mxnet, создана 3х-слойная сеть с функцией активации relu и 128 нейронами на входном слое. Скорость обучения – 0.1, обучение произведено для 40 эпох. Проверка точности производилась при разбиении на k_folds = 5.






```
Test loss: 0.114516
5d-fold validation: Avg train loss: 0.09869133830070495, Avg test loss: 0.12053434103727341
```

Применение нейросети для выборки, обработанной на стадии EDA, позволяет улучшить рейтинг на 227 позиций по сравнению с применением к исходному датасету.