Data Science Initiative
BROWN

Final Project Presentation

—— OYO rental price prediction in China

Brown University, Data Science Initiative,22fall

GitHub: https://github.com/AstrosiosaurQ7/data1030_final_project.git

Xiaoyan Liu

06/12/2022

Content

1. Recap
2. Cross validation
3. Results
4. Improve

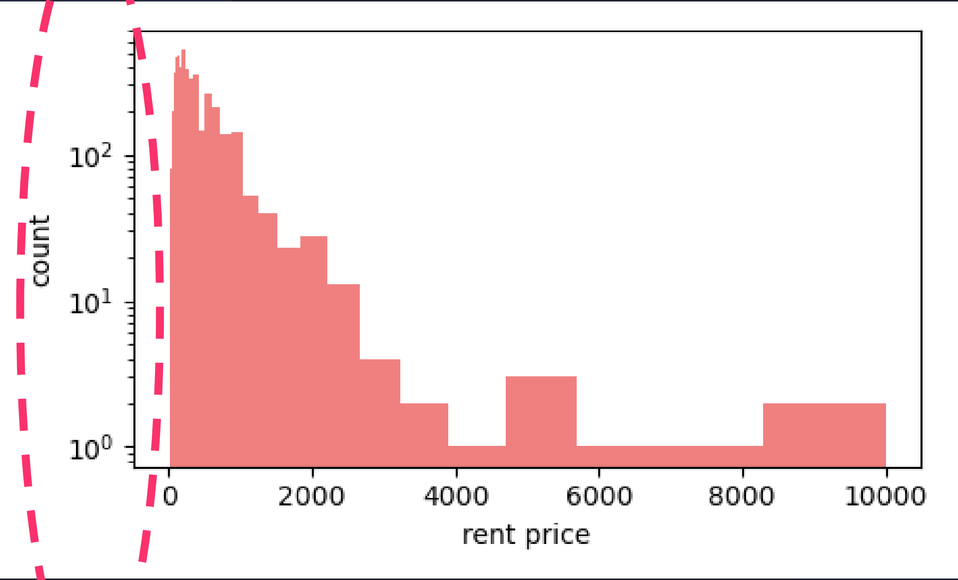
01 Target variable

Problem

Target variable ： rental price / $

Regression / Right-skewed

Feature matrix shape： (5834,25)



Data Resource

OYO APP ， Chinese OTA platform : XieCheng / MeiTuan/FeiZhu

Kaggle [Oyo Rental Price Prediction in China | Kaggle]

➢ Enlarge the coordinate axis by log function

- This report hopes to predict the rental price of OYO hotels according to the property type 、 hotel location and so on.

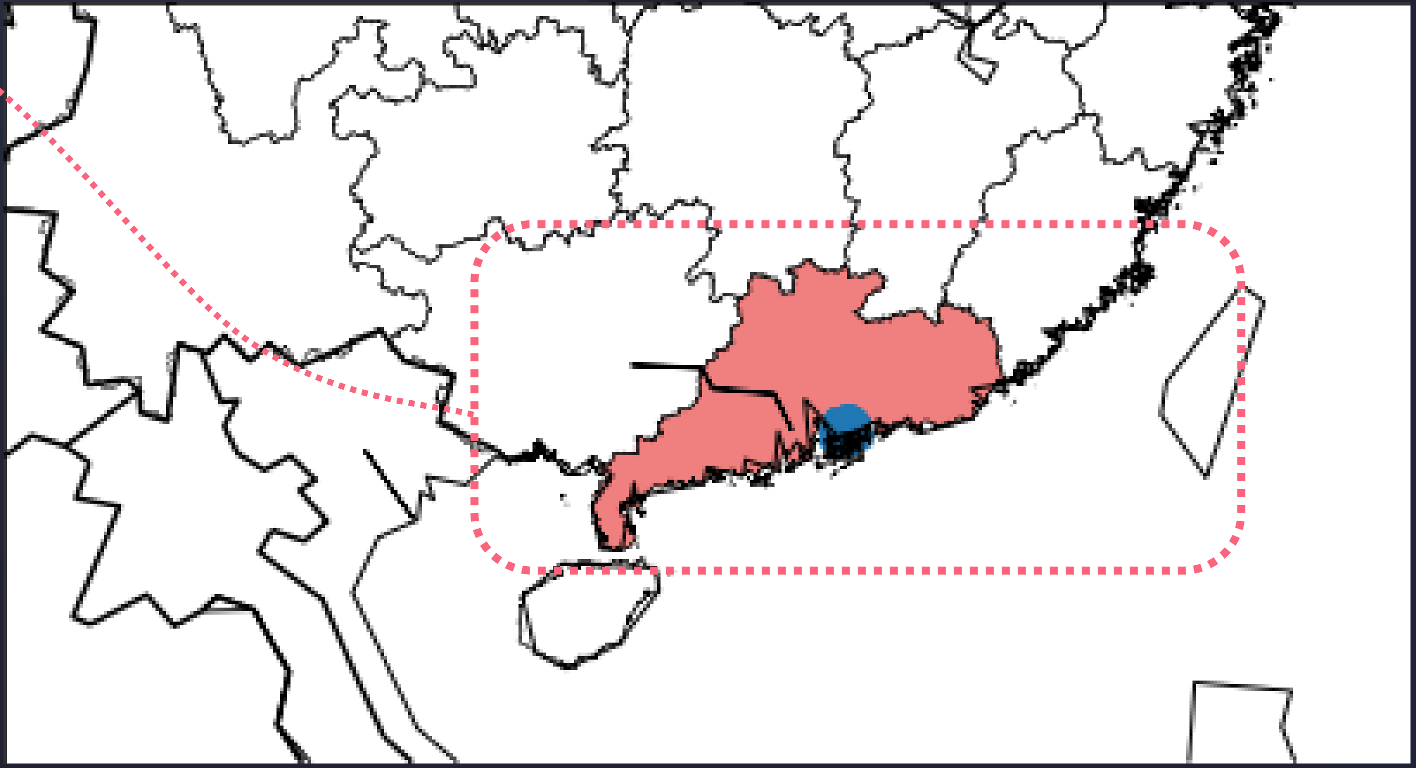
01 EDA-Location features

location

```
df[['longitude(East)','latitude(North)']]
```

	longitude(East)	latitude(North)
0	114.059600	22.542900
1	114.043225	22.539490
2	114.079426	22.508573
3	114.079035	22.508697
4	114.055590	22.509502
...	...	...
5829	114.194588	22.619618
5830	114.201327	22.606116

Most hotels are concentrated in the surrounding cities of Shenzhen and Hong Kong, which are near to the seaport and have developed tourism and economic industries.



01 Preprocessing

preprocessing

Step 1. drop worthless features : amenities, has_availability

Step 2. preprocessor encoding

- One-Hot encoder : 'bed_type','host_is_superhost', 'instant_bookable', 'room_type'

- Ordinal encoder : 'cancellation_policy','property_type'

- MinMax Scaler: 'accommodates','availability_30' , 'calculated_host_listings_count' , 'guests_included' , 'number_of_reviews','bathrooms', 'bedrooms', 'beds' , 'host_listings_count' , 'review_scores_checkin', 'review_scores_communication' , 'review_scores_location' , 'review_scores_rating', 'review_scores_value'

- Standard Scaler : 'maximum_nights'

02 Cross validation

split & pipeline

def MLpipe_KFold_RMSE

- Split 20% data to be test set
- Used **Kfold (n_splits=4,shuffle=True)**
- GridSearchCV function
- evaluation metric= **RMSE**
- Output best parameters and best score
- Save the scores and y_test for the baseline model

```
def MLpipe_KFold_RMSE(X,y,preprocessor,reg,param_grid):
    test_scores = np.zeros(nr_states)
    for i in range(nr_states):

        X_other, X_test, y_other, y_test = train_test_split(X,y,test_size = 0.2,random_state=42*i)
        kf = KFold(n_splits=4,shuffle=True,random_state=42*i)
        pipe = make_pipeline(preprocessor,reg)
        grid = GridSearchCV(pipe, param_grid=param_grid,scoring = 'neg_mean_squared_error',
                        cv=kf, return_train_score = True, n_jobs=-1, verbose=True)
        grid.fit(X_other, y_other)
        results = pd.DataFrame(grid.cv_results_)

        print('best model parameters:',grid.best_params_)
        print('validation score:',grid.best_score_)
    # save the model
        final_models.append(grid)
    # calculate and save the test score
        y_test_pred = final_models[-1].predict(X_test)
        test_scores[i] = np.sqrt(mean_squared_error(y_test,y_test_pred))
        print('RMSE test score:',test_scores[i])
    return test_scores,y_test
```

02 **Cross validation**

ML algo & tune

L1 regularized linear regression(Lasso)

✓ alpha : 0.25, 2.5, 25

✓ random_state=42

RandomForestRegressor

✓ max_features : [3, 5, 7, 9]
✓ max_depth : [3, 5, 7, 9]

✓ random_state=42

SVR (Support Vector)

✓ gamma: [0.1, 10, 100]
✓ C : [0.1, 1, 10]

✓ kernel='rbf'

KNeighborsRegressor(

✓ n_neighbors': [5, 25, 50]

✓ weights='uniform’

03 Results

scores

+

L1 :

- test score: [332.50481953, 321.1485038 , 249.33466631, 369.32089209,298.27287924, 349.18486624, 364.13289666, 258.50704218,229.58742343, 337.10134802]
- test scores mean : 310.9095337515326
- test scores standard deviation : 47.198

SVR :

- test score: [457.0634567, 439.1388, 352.7975, 502.4557654, 409.218457, 442.50976546yy, 495.1113467, 357.6800, 299.077, 460.71044]
- test scores mean : 421.576384
- test scores standard deviation : 62.8157611

RandomForestRegressor :

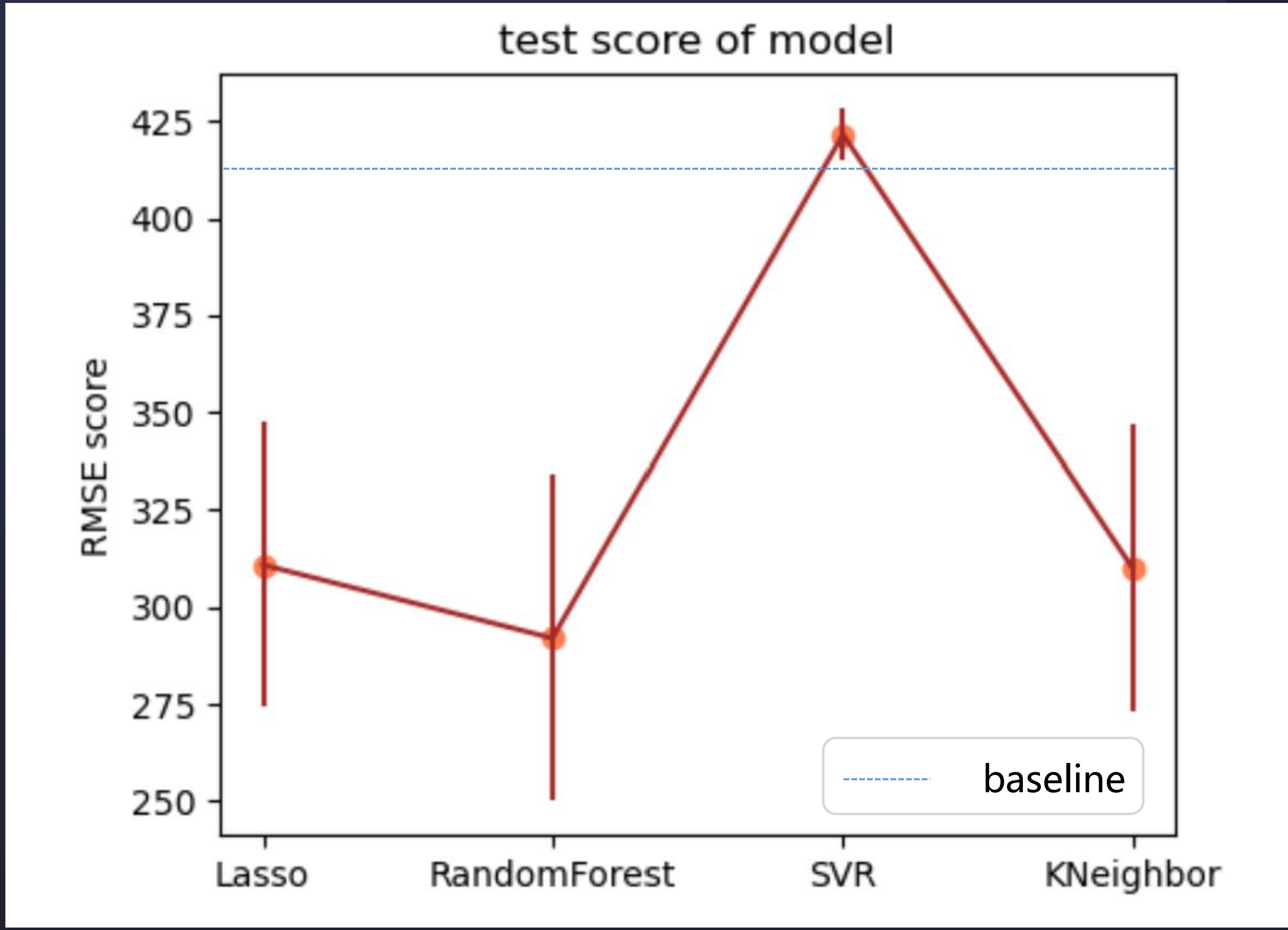
- test score: [307.10369093, 305.09515619, 215.86303583, 344.76157886,292.349, 336.76020741, 360.37117433, 239.6237,218.4528, 299.79961715]
- test scores mean : 292.01818453
- test scores standard deviation : 48.858289905

KNeighborsRegressor :

- test score: [318.20838074, 320.60138962, 265.89838554, 371.5695846 ,289.9438, 335.33792052, 384.54992622, 252.862433 , 222.87348307, 338.97971562]
- test scores mean : 310.082503
- test scores standard deviation : 49.206679

03 Results

Best model



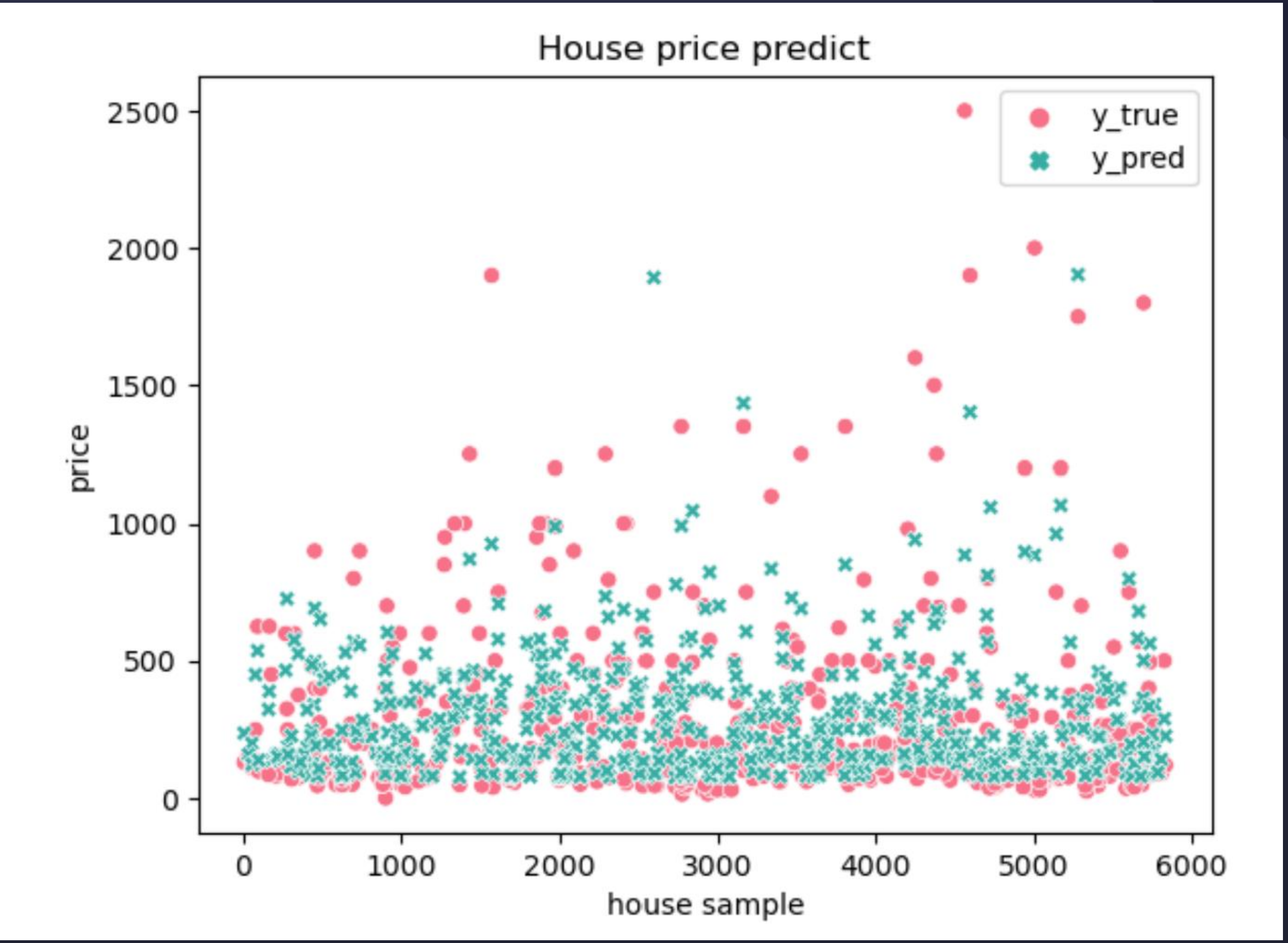
- baseline RMSE: 415.7852569807543
  (y_mean=y_pred)

- test scores mean : 333.64
- test scores standard deviation :73.346

- best model parameters: {'randomforestregressor__max_depth': 9, 'randomforestregressor__max_features': 5}

03 Results

Scatter true & pred



- best model parameters: {'max_depth': 9, 'max_features': 5}

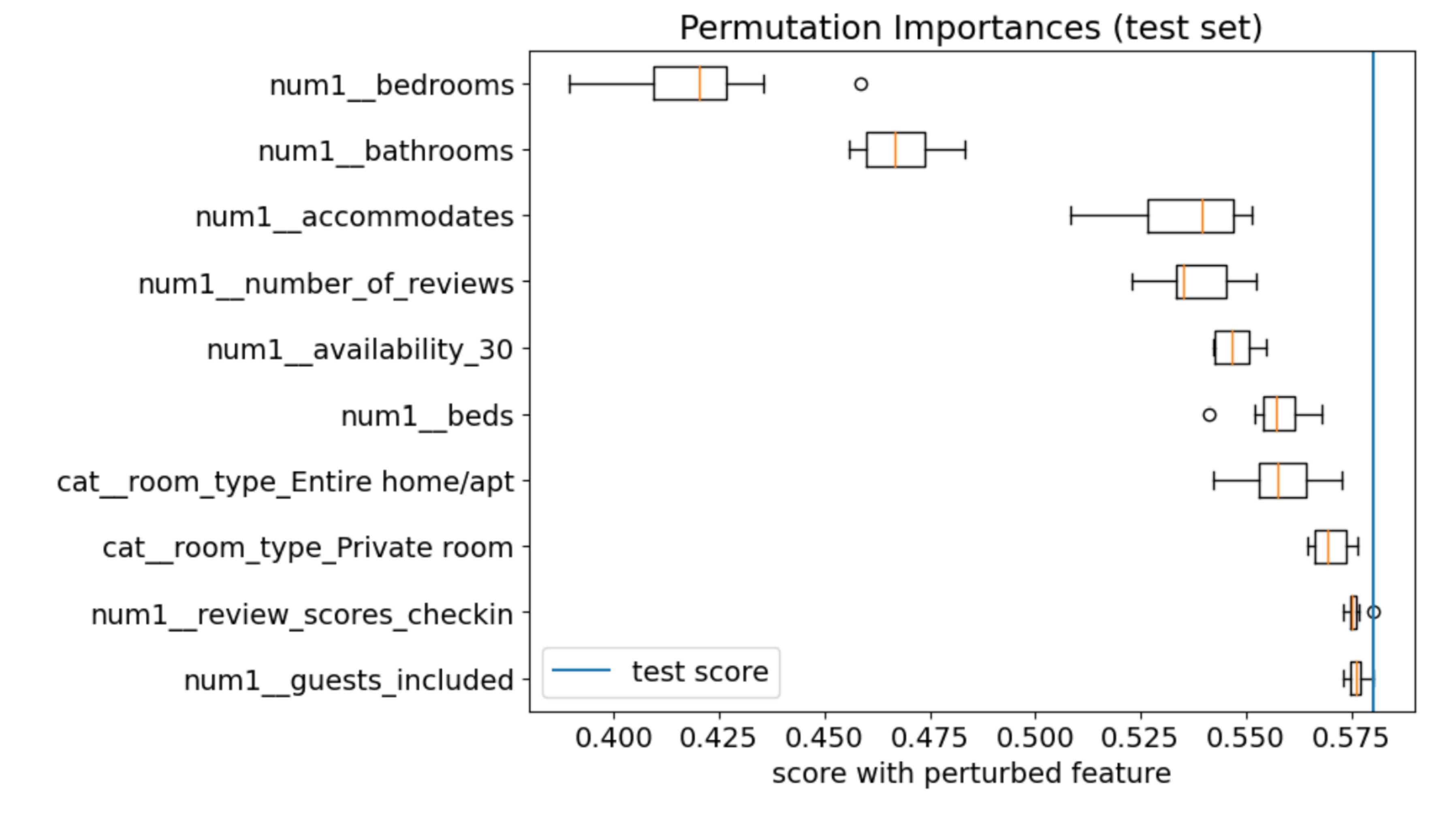
- train set shape : (4667, 30)
- test set shape : (583, 30)

- running time 0.3580458164215088

- test scores 202.72660072854498

03 Results

global feature

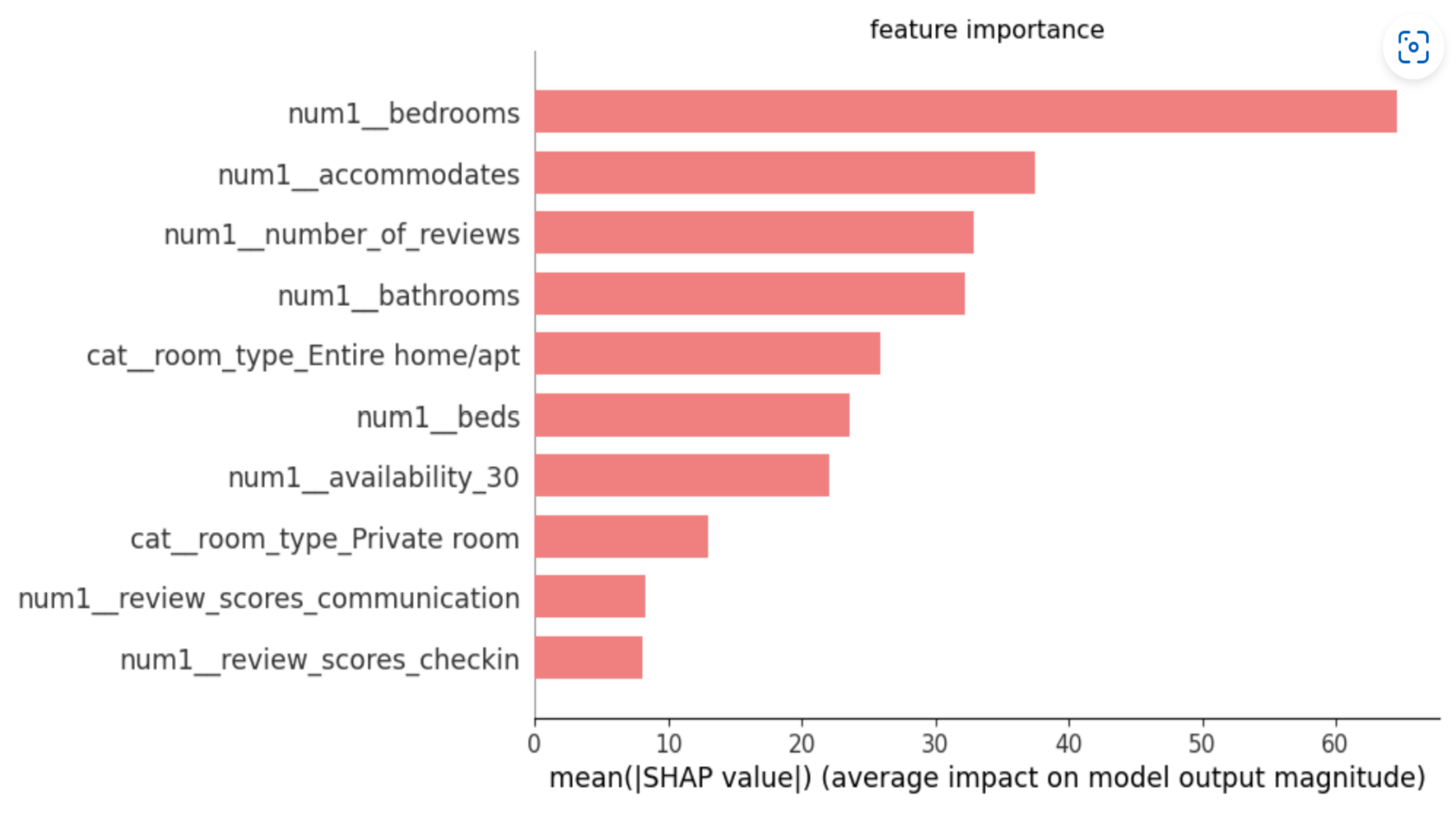


Shuffling

TOP 10

03 Results

global feature

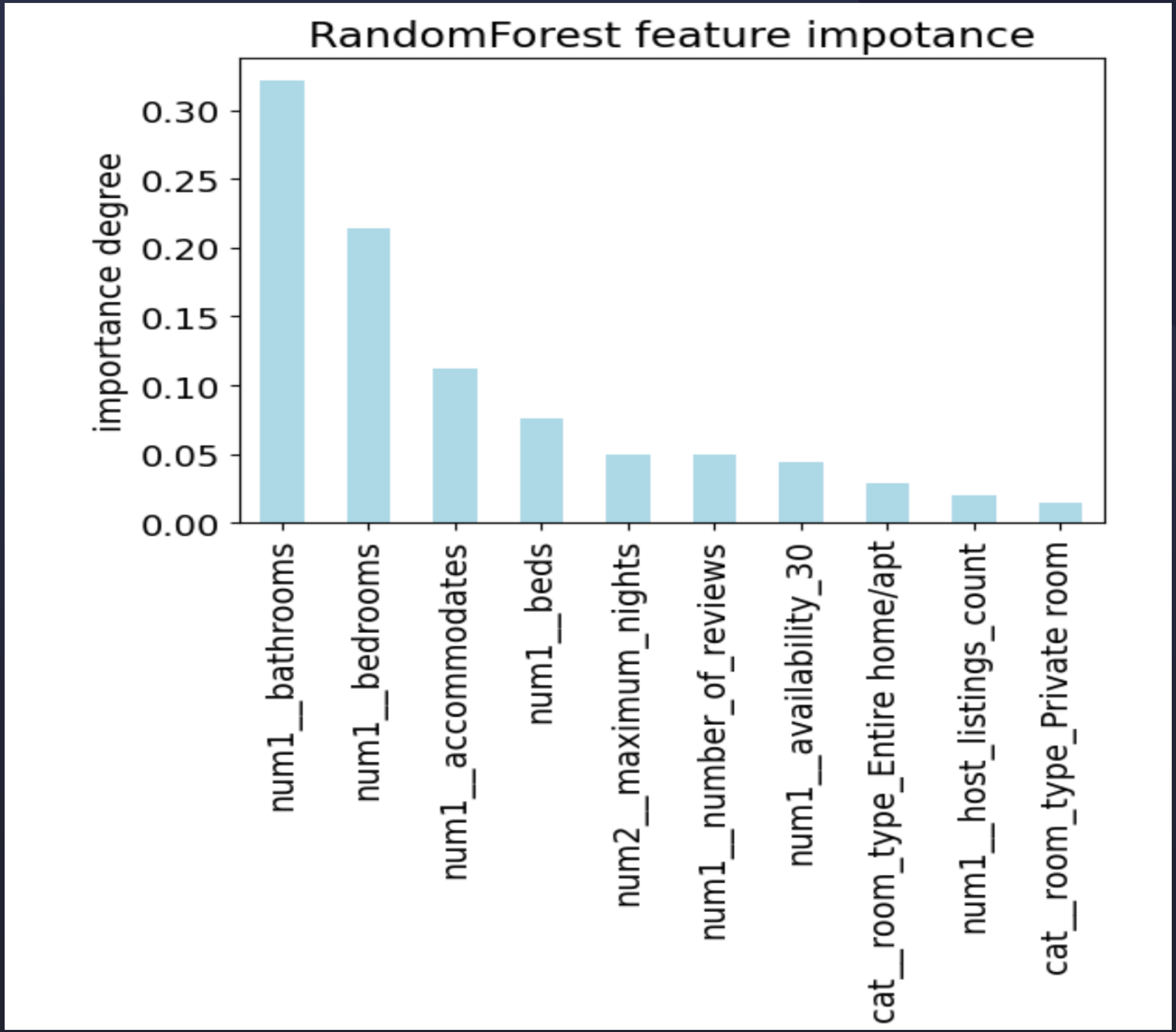


SHAP Global

TOP 10

03 Results

global feature



Randomforest feature_importances function

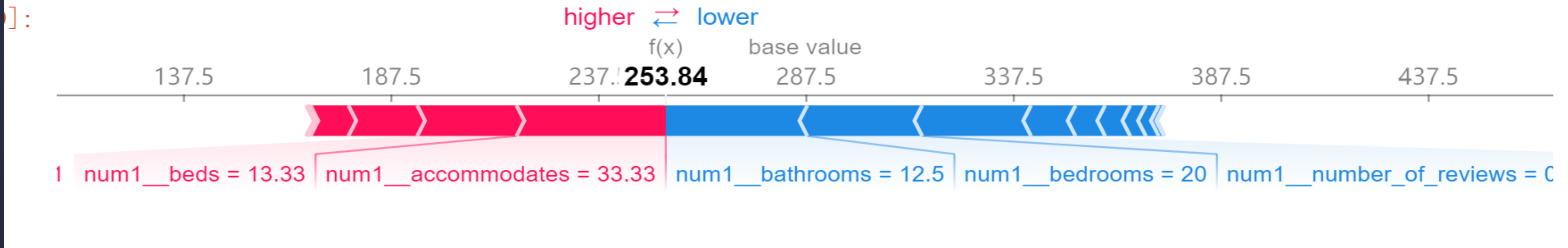
- Those results are very similar to the previous , but the sort is different.

03 Results

local feature

Point 0

```
#point 0
shap.force_plot(explainer.expected_value, shap_values[0, :], df_test.iloc[0, :])
```

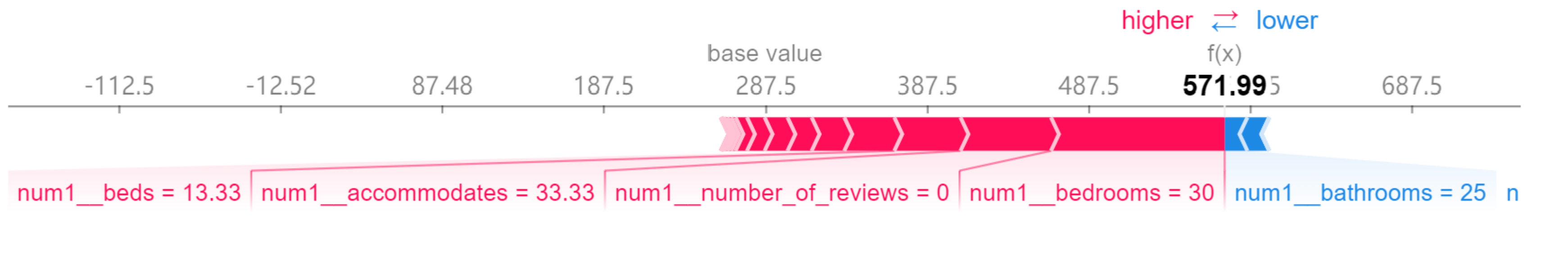


Point 20

[90]:

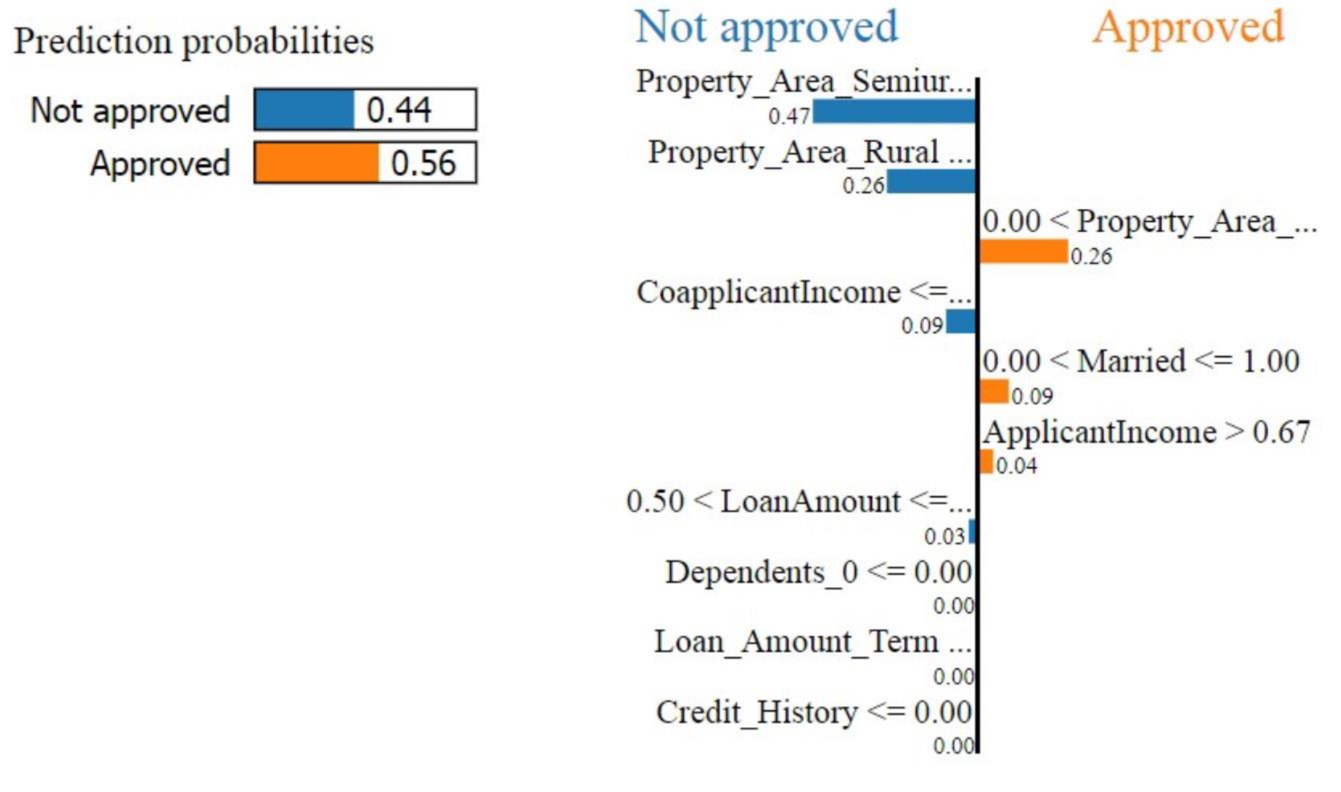
```
#point
shap.force_plot(explainer.expected_value, shap_values[20, :], df_test.iloc[20, :])
```

Out[90]:



04 Improve

improve



- XGBoost

- SHAP, LIME, and Anchors, provide local, model-agnostic interpretability methods.

Reference:
https://towardsdatascience.com/three-interpretability-methods-to-consider-when-developing-your-machine-learning-model-5bf368b47fac
https://christophm.github.io/interpretable-ml-book/interpretability.html

Data Science Initiative
BROWN

THANK YOU FOR WATCHING