Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Домашнее Задание №1
по дисциплине
«Методы машинного обучения»
на тему

«Прогноз цены на Airbnb»

Выполнил:
студент группы ИУ5-21М
Чжан Чжибо

Москва — 2021 г.

1. Задаче обучения

Прогноз цены комнаты на airbnb(airbnb-это платформа для онлайн бронирования комнаты) на основе данных о 50000 номеров в Нью-Йорке.

2. Обучение с полной предобработкой данных

Подключим все необходимые библиотеки:

In [1]:
```
from sklearn import preprocessing, metrics
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="ticks", color_codes=True)
```

Загрузим данные:

In [2]:
```
dataset = pd.read_csv(r"D:/DATA/AB_NYC_2019.csv")
```

In [3]:
```
dataset.head(5)
```

Просмотр данных:

In [3]:
```
dataset.head(5)
```

Out[3]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_revi
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	

In [3]:
```
dataset.head(5)
```

Out[3]:

ood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listings_count	availability_365
gton	40.64749	-73.97237	Private room	149	1	9	2018-10-19	0.21	6	365
own	40.75362	-73.98377	Entire home/apt	225	1	45	2019-05-21	0.38	2	355
lem	40.80902	-73.94190	Private room	150	3	0	NaN	NaN	1	365
Hill	40.68514	-73.95976	Entire home/apt	89	1	270	2019-07-05	4.64	1	194
lem	40.79851	-73.94399	Entire home/apt	80	10	9	2018-11-19	0.10	1	0

Просмотр пропусков в данных :

```
In [4]: dataset.isnull().sum()
```

```
Out[4]: id                                0
        name                             16
        host_id                           0
        host_name                        21
        neighbourhood_group               0
        neighbourhood                     0
        latitude                          0
        longitude                         0
        room_type                         0
        price                             0
        minimum_nights                    0
        number_of_reviews                 0
        last_review                   10052
        reviews_per_month             10052
        calculated_host_listings_count    0
        availability_365                  0
        dtype: int64
```

Заполнение пропусков:

```
In [5]: dataset.fillna({'reviews_per_month':0}, inplace=True) #填空值
        dataset.fillna({'name':"noname"}, inplace=True)
        dataset.fillna({'host_name':"nohostname"}, inplace=True)
        dataset.fillna({'last_review':"notreviewed"}, inplace=True)
```

```
In [6]: dataset.isnull().sum()
```

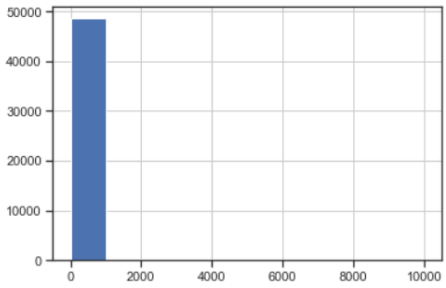
```
Out[6]: id                                0
        name                              0
        host_id                           0
        host_name                         0
        neighbourhood_group               0
        neighbourhood                     0
        latitude                          0
        longitude                         0
        room_type                         0
        price                             0
        minimum_nights                    0
        number_of_reviews                 0
        last_review                       0
        reviews_per_month                 0
        calculated_host_listings_count    0
        availability_365                  0
        dtype: int64
```

Просмотр цены:

```
In [7]: dataset["price"].describe()
```

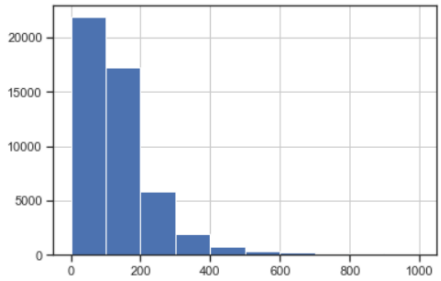
```
Out[7]: count    48895.000000
        mean       152.720687
        std        240.154170
        min          0.000000
        25%         69.000000
        50%        106.000000
        75%        175.000000
        max      10000.000000
        Name: price, dtype: float64
```

```
In [8]: hist_price=dataset["price"].hist()
```



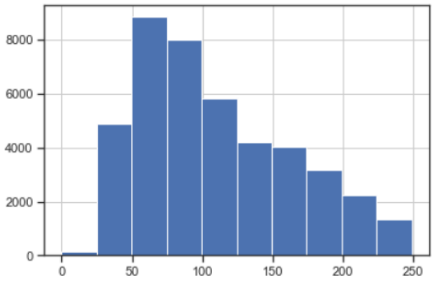
Увидим что большинство данных цены меньше 1000, посмотрим на цены меньше 1000:

```
In [9]: hist_price1=dataset["price"][dataset["price"]<1000].hist()
```



Увидим что большинство данных цены меньше 250, посмотрим на цены меньше 250:

```
In [10]: hist_price2=dataset["price"][dataset["price"]<250].hist()
```



Устранение цены, которые больше 250:

```
In [11]: dataset=dataset[dataset["price"]<250]
```

Проверка существует ли в данных повторные ряды:

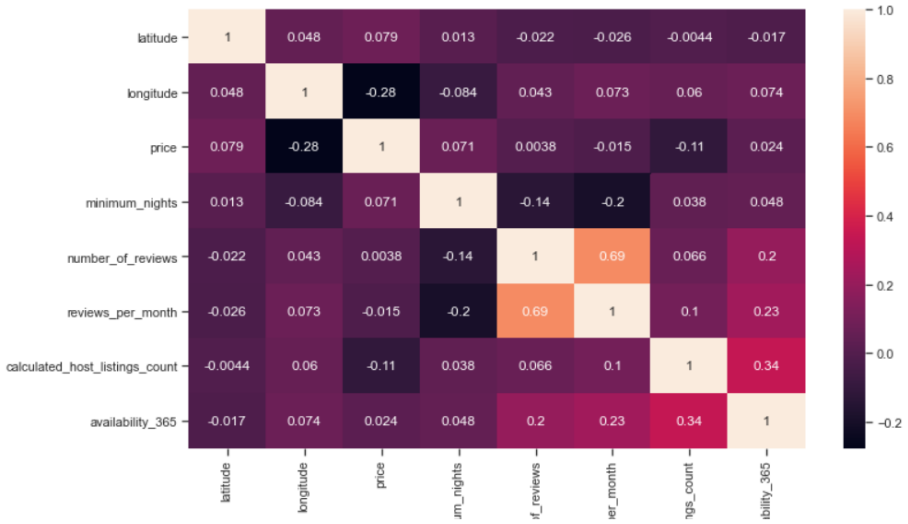
```
In [12]: dataset.duplicated().sum()
Out[12]: 0
```

Устранение ненужных признаков:

```
In [13]: dataset.drop(['name','id','host_id','last_review','host_name','neighbourhood'], axis = 1, inplace = True)
```

Посмотрим коррелятивность между признаками:

```
In [14]: corr = dataset.corr(method='kendall')
         plt.figure(figsize=(12,7))
         sns.heatmap(corr, annot=True)
         dataset.columns
Out[14]: Index(['neighbourhood_group', 'latitude', 'longitude', 'room_type', 'price',
                'minimum_nights', 'number_of_reviews', 'reviews_per_month',
                'calculated_host_listings_count', 'availability_365'],
               dtype='object')
```



Увидим что между признаками нет сильных коррелятивноси

Разделение данных на X и y:

```
In [15]: y=dataset['price']
         X=dataset.drop(['price'],axis=1)
```

Кодирование категориальных признаков:

```
In [16]: X = pd.get_dummies(X, prefix=['neighbourhood_group', 'room_type'], drop_first=True)
```

Разделение данных на тестовый и обучающий:

```
In [17]: X_train, X_test, y_train, y_test =  train_test_split(X, y, test_size = 0.2, random_state= 0)
```

Просмотр размерности данных:

In [18]:
```
print('Dimensions of the training feature matrix: {}'.format(X_train.shape))
print('Dimensions of the training target vector: {}'.format(y_train.shape))
print('Dimensions of the test feature matrix: {}'.format(X_test.shape))
print('Dimensions of the test target vector: {}'.format(y_test.shape))
```

```
Dimensions of the training feature matrix: (34135, 13)
Dimensions of the training target vector: (34135,)
Dimensions of the test feature matrix: (8534, 13)
Dimensions of the test target vector: (8534,)
```

Масштабирование признаков:

In [19]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

Выбор лучших параметров для модели:

In [30]:
```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.model_selection import GridSearchCV
param_grid = [{'n_estimators': [50,500,900], 'learning_rate': [0.01,0.1,1]}]
gbreg=GradientBoostingRegressor()
grid_search = GridSearchCV(gbreg, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)
grid_search.fit(X_train,y_train)
```

Out[30]:
```
GridSearchCV(cv=5, estimator=GradientBoostingRegressor(),
             param_grid=[{'learning_rate': [0.01, 0.1, 1],
                          'n_estimators': [50, 500, 900]}],
             return_train_score=True, scoring='neg_mean_squared_error')
```

Лучшие параметры:

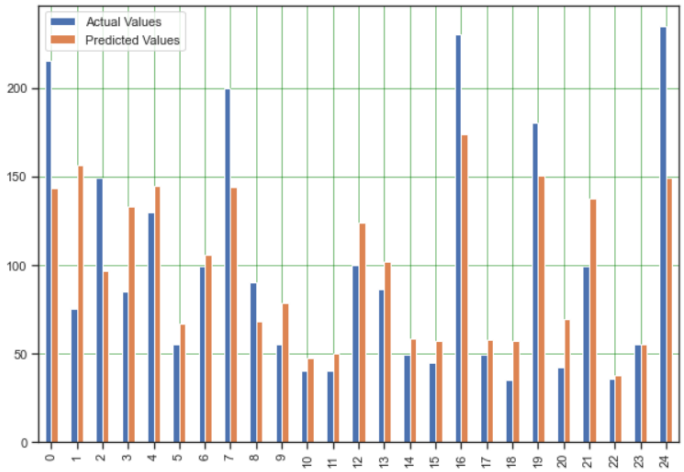
In [32]:
```
grid_search.best_params_
```

Out[32]:
```
{'learning_rate': 0.1, 'n_estimators': 900}
```

Обучение и оценка:

In [34]:
```
model1=grid_search.best_estimator_
y_pred=model1.predict(X_test)

from sklearn.metrics import r2_score
print("R2 score: ",r2_score(y_test,y_pred))
print("RMSE: ",np.sqrt(mean_squared_error(y_test,y_pred)))

error_diff = pd.DataFrame({'Actual Values': np.array(y_test).flatten(), 'Predicted Values': y_pred.flatten()})
print(error_diff.head(5))

#Visualize the error
df1 = error_diff.head(25)
df1.plot(kind='bar',figsize=(10,7))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```

```
R2 score:  0.5815441396198857
RMSE:  34.99801362335561
   Actual Values  Predicted Values
0            215        143.225360
1             75        156.504707
2            149         96.622063
3             85        132.936577
4            130        144.926863
```



3. Обучение, которое содержит только минимальную предобработку данных, необходимую для построения модели

Загрузим данные:

```
In [57]: dataset2 = pd.read_csv(r"D:/DATA/AB_NYC_2019.csv")
```

Заполнение пропусков:

```
In [58]: dataset2.fillna({'reviews_per_month':0}, inplace=True) #填空值
         dataset2.fillna({'name':"noname"}, inplace=True)
         dataset2.fillna({'host_name':"nohostname"}, inplace=True)
         dataset2.fillna({'last_review':"notreviewed"}, inplace=True)
```

Устранение ненужных признаков:

```
In [59]: dataset2.drop(['name','id','host_id','last_review','host_name','neighbourhood'], axis = 1, inplace = True)
```

Разделение данных на X и y:

```
In [60]: y2=dataset2['price']
         X2=dataset2.drop(['price'],axis=1)
```

Кодирование категориальных признаков:

```
In [61]: X2 = pd.get_dummies(X2, prefix=['neighbourhood_group', 'room_type'], drop_first=True)
```

Разделение данных на тестовый и обучающий:

```
In [62]: X_train2, X_test2, y_train2, y_test2 =  train_test_split(X2,y2,test_size = 0.2, random_state= 0)
```

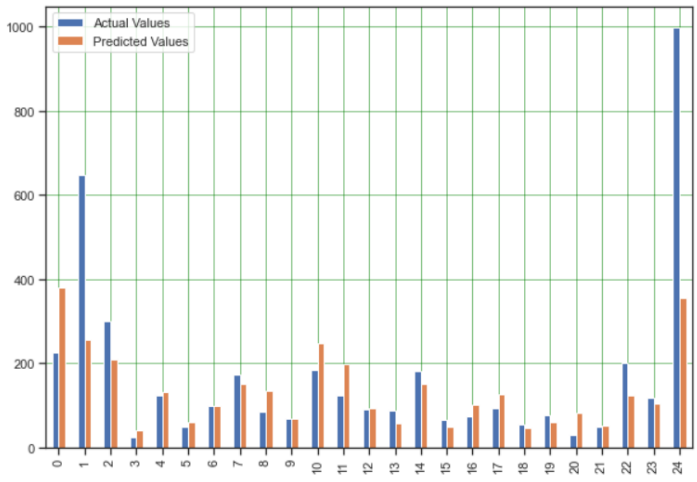
In [63]:
```
print('Dimensions of the training feature matrix: {}'.format(X_train2.shape))
print('Dimensions of the training target vector: {}'.format(y_train2.shape))
print('Dimensions of the test feature matrix: {}'.format(X_test2.shape))
print('Dimensions of the test target vector: {}'.format(y_test2.shape))
```

```
Dimensions of the training feature matrix: (39116, 13)
Dimensions of the training target vector: (39116,)
Dimensions of the test feature matrix: (9779, 13)
Dimensions of the test target vector: (9779,)
```

Обучение и оценка:

In [65]:
```
model2 = GradientBoostingRegressor(n_estimators=900, learning_rate=0.1)
model2.fit(X_train2,y_train2)

y_pred2=model2.predict(X_test2)

from sklearn.metrics import r2_score
print("R2 score: ",r2_score(y_test2,y_pred2))
print("RMSE: ",np.sqrt(mean_squared_error(y_test2,y_pred2)))

error_diff = pd.DataFrame({'Actual Values': np.array(y_test2).flatten(), 'Predicted Values': y_pred2.flatten()})
print(error_diff.head(5))

df1 = error_diff.head(25)
df1.plot(kind='bar',figsize=(10,7))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```

```
R2 score:  -0.25577749545527184
RMSE:  262.89879252672864
   Actual Values  Predicted Values
0            225        379.675201
1            649        256.961130
2            300        208.642989
3             26         41.508656
4            125        133.213625
```



Увидим, что после удаления этапов обработки, таких как удаление ненормальных величины и масштабирование функций, оценка R2 и RMSE значительно хуже чем оценки обучения с полными этапами обработки.

4. Обучение с использованием библиотеки TPOT и сравнение оценок

TPOT - это инструмент автоматического машинного обучения Python, основанный на генетических алгоритмах для оптимизации трубопроводов (pipeline)машинного обучения.

Он может исследовать тысячи возможных pipeline, найти лучший pipeline для набора данных и выполнить полную подгонку модели.

При определении модели указаны параметры:

generations(Количество итераций)—5

population_size(Количество лиц, оставшихся в каждом поколении наследования)—20

verbosity(информация, отображаемая во время работы)—2(информация содержит индикатор выполнения и текущего поколения)

scoring(мера оценка)—R2

Если используем данные с полными этапами обработки, как вовтором разделе:

```
In [74]: from tpot import TPOTRegressor
         model3=TPOTRegressor(generations=5,population_size=20,verbosity=2,scoring='r2')
         model3.fit(X_train,y_train)
         print(model3.score(X_test,y_test))

         Generation 1 - Current best internal CV score: 0.6060210358243773

         Generation 2 - Current best internal CV score: 0.6060210358243773

         Generation 3 - Current best internal CV score: 0.6060210358243773

         Generation 4 - Current best internal CV score: 0.6083860686785301

         Generation 5 - Current best internal CV score: 0.608915064504387

         Best pipeline: RandomForestRegressor(input_matrix, bootstrap=True, max_features=0.5, min_samples_leaf=4, min_samples_split=20, n_estimat
         ors=100)
         0.6001098420311461
```

Увидим что обучение с использованием библиотеки TPOT получил чуть чуть лучше оценку R2, чем обучение во втором разделе, но эта автоматическая модель отнимала намного больше времени чем моднль во втором разделе.

Если используем данные с только необходимыми этапами обработки, как в третьем разделе:

In [75]:
```
from tpot import TPOTRegressor
model3=TPOTRegressor(generations=5,population_size=20,verbosity=2,scoring='r2')
model3.fit(X_train2,y_train2)
print(model3.score(X_test2,y_test2))
```

```
Generation 1 - Current best internal CV score: 0.18682790327212134

Generation 2 - Current best internal CV score: 0.18682790327212134

Generation 3 - Current best internal CV score: 0.18682790327212134

Generation 4 - Current best internal CV score: 0.18682790327212134

Generation 5 - Current best internal CV score: 0.18682790327212134

Best pipeline: RandomForestRegressor(input_matrix, bootstrap=True, max_features=0.8500000000000001, min_samples_leaf=3, min_samples_spli
t=6, n_estimators=100)
0.15006458402397094
```

Увидим что обучение с использованием библиотеки TPOT получил намного лучше оценку R2, чем обучение в третьем разделе, но эта оценка ещё значительно хуже чем обучение во втором разделе.Эта автоматическая модель всегда отнимала намного больше времени чем модели во втором разделе и третьем разделе.

Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Разведочный анализ данных. Исследование и визуализация данных» [Электронный ресурс] // GitHub. — 2019. — Режим доступа: https://github.com/ugapanyuk/ml_course/wiki/LAB_EDA_VISUALIZATION (дата обращения: 13.02.2019)

[2] https://www.kaggle.com/datasets