

Курсовой проект

Цель:

Предсказать стоимость квартиры. Зачем?

В банках, страховых компаниях: узнать истинную стоимость имущества (залога). Принять решение о выдаче ипотеки/страховки. Принять решение о проценте по ипотеке/страховки. На площадках объявлений (Авито, Циан, ...), найти недооцененные квартиры (выгодные предложения, показать их пользователям, продемонстрировать пользователям рыночную стоимость квартиры).

Для тех кто продаёт рекомендовать цену продажи.

Для инвесторов готовых вкладываться в квартиры, определять рыночную стоимость квартир, производить поиск недооцененных активов и вести торговлю этими активами на рынке недвижимости.

Исходные данные:

Данные для дальнейших расчётов были взяты отсюда:

<https://www.kaggle.com/c/real-estate-price-prediction-moscow>

Предобработка данных.

Для начала полученные данные были прочитаны и просмотрены типы данных, а также чтобы каждый файл с данными открывался без проблем и выводил на экран свои основные характеристики: `df.shape`, `df.dtypes`, `df.info()`, `df.describe`.

Далее был произведен первичный анализ данных при запросе команды `df.head()`, начали всплывать пропуски в данных.

Далее был проведён более тщательный анализ данных, при котором было выяснено, что в данных были обнаружены следующие нюансы.

1. Пропуски.

2. Нулевые значения, которых не должно быть по определению данного столбца и в реальных условиях их невозможно использовать.

3. Выбросы.

После проведения анализа всех этих нюансов, было принято решение заменять все неувязки в данных в каждом столбце на медиану. Так как она устойчива по определению к выбросам и где было незначительное количество, отсортированных нулевых значений, которых там не должно присутствовать. Они все были заменены на медиану.

После этого когда был выверен файл с `train` данными, аналогичные операции проводились над файлом `test`.

Обучение модели.

Далее приступил к самому обучению модели. Модель была выбрана `RandomForestRegressor`. Далее после того как определился с моделью обучения стал через неё прогонять данные из файла `train`, когда получил результат $R^2 = 0.7068$. Затем стал прогонять через данную модель `test` и получил $R^2 = 0.8068$.

Расчёты.

Ниже прилагаю код из интерактивного редактора кода.

Использую его по двум причинам: во-первых удобно проверять код на ошибки, а во-вторых данный вариант менее ресурсоёмкий, чем стандартные программы для обработки кода. Для подтверждения этого же кода были созданы два `Python_script`, содержащие в себе вариант `train` и `test`.

Python 3.8.10 (default, Jun 2 2021, 10:49:15)

[GCC 9.4.0] on linux

Type "help", "copyright", "credits" or "license" for more information.

```
>>> import numpy as np
```

```
>>> import pandas as pd>>> import scipy
```

```
>>> import sklearn
```

```
>>> import matplotlib.pyplot as plt
```

```
>>> import seaborn as sns
```

```

>>> import random
>>> from sklearn.model_selection import train_test_split, cross_val_score
>>> from sklearn.preprocessing import StandardScaler, MinMaxScaler
>>> from sklearn.ensemble import RandomForestRegressor
>>> from sklearn.metrics import r2_score as r2
>>> from sklearn.model_selection import KFold, GridSearchCV
>>> df = pd.read_csv('train.csv')
>>> df.head()
Id DistrictId Rooms Square ... Helthcare_2 Shops_1 Shops_2
Price
0 11809
27
3 115 ...
1
3 True 305018.871089
1 3013
22
1
39 ...
1
3 True 177734.553407
2 8215
1
3
78 ...
3
1 True 282078.720850
3 2352
1
1
40 ...
1
1 True 168106.007630
4 13866
94
2
64 ...
1
6 True 343995.102962
[5 rows x 20 columns]
>>> df.shape
(10000, 20)
>>> df.dtypes
Id
int64
DistrictId
int64
Rooms
int64
Square
int64
LifeSquare

```

```
int64
KitchenSquare
int64
Floor
int64
HouseFloor
float64
HouseYear
int64
Ecology_1
float64
Ecology_2
bool
Ecology_3
bool
Social_1
int64
Social_2
int64
Social_3
int64
Healthcare_1 float64
Helthcare_2
int64
Shops_1
int64
Shops_2
bool
Price
float64
dtype: object
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 20 columns):
Id
10000 non-null int64
DistrictId
10000 non-null int64
Rooms
10000 non-null int64
Square
10000 non-null int64LifeSquare
10000 non-null int64
KitchenSquare 10000 non-null int64
Floor
10000 non-null int64
HouseFloor
10000 non-null float64
HouseYear
10000 non-null int64
Ecology_1
```

```

10000 non-null float64
Ecology_2
10000 non-null bool
Ecology_3
10000 non-null bool
Social_1
10000 non-null int64
Social_2
10000 non-null int64
Social_3
10000 non-null int64
Healthcare_1 10000 non-null float64
Helthcare_2
10000 non-null int64
Shops_1
10000 non-null int64
Shops_2
10000 non-null bool
Price
10000 non-null float64
dtypes: bool(3), float64(4), int64(13)
memory usage: 1.3 MB
>>> X = df.iloc[:, 0:19]
>>> y = df.iloc[:, 19]
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
>>> sc = StandardScaler()
>>> X_train = sc.fit_transform(X_train)
>>> X_test = sc.transform(X_test)
>>> regressor = RandomForestRegressor(n_estimators=20, random_state=42)
>>> regressor.fit(X_train, y_train)
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
max_depth=None, max_features='auto', max_leaf_nodes=None,
max_samples=None, min_impurity_decrease=0.0,
min_impurity_split=None, min_samples_leaf=1,
min_samples_split=2, min_weight_fraction_leaf=0.0,
n_estimators=20, n_jobs=None, oob_score=False,
random_state=42, verbose=0, warm_start=False)
>>> y_pred = regressor.predict(X_test)
>>>
>>> r2(y_test, y_pred)
0.7068526907254846
>>>
>>> import numpy as np
>>> import pandas as pd
>>> import scipy
>>> import sklearn
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
>>> import random
>>> from sklearn.model_selection import train_test_split, cross_val_score
>>> from sklearn.preprocessing import StandardScaler, MinMaxScaler
>>> from sklearn.ensemble import RandomForestRegressor

```

```

>>> from sklearn.metrics import r2_score as r2
>>> from sklearn.model_selection import KFold, GridSearchCV
>>> df = pd.read_csv('test.csv')
>>> df.head()
0
1
2
3
4
Id DistrictId Rooms Square ... Helthcare_2 Shops_1 Shops_2
4567
44
1
36 ...
1
1 True 63.323353
5925
62
1
42 ...
1
3 True 63.323353
960
27
2
59 ...
1
1 True 63.323353
3848
23
3
49 ...
1
3 True 63.323353
746
74
1
53 ...
1
6 True 63.323353
[5 rows x 20 columns]
>>> df.shape
(5000, 20)
>>> df.dtypes
Id
int64
DistrictId
int64
Rooms
int64
Square
int64
LifeSquare

```

```
int64
KitchenSquare
int64
Floor
int64
HouseFloor
float64
HouseYear
int64
Ecology_1
float64
Ecology_2
bool
Ecology_3
bool
Social_1
int64
Social_2
int64
Social_3
int64
Healthcare_1 float64
Helthcare_2
int64
Shops_1
int64
Shops_2
bool
Price
float64
dtype: object
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 20 columns):
Id
5000 non-null int64
DistrictId
5000 non-null int64
Rooms
5000 non-null int64
Square
5000 non-null int64
LifeSquare
5000 non-null int64
KitchenSquare 5000 non-null int64
Floor
5000 non-null int64
HouseFloor
5000 non-null float64
HouseYear
5000 non-null int64
```

```

Ecology_1
5000 non-null float64
Ecology_2
5000 non-null bool
Ecology_3
5000 non-null bool
Social_1
5000 non-null int64
Social_2
5000 non-null int64
Social_3
5000 non-null int64
Healthcare_1 5000 non-null float64
PriceHelthcare_2
5000 non-null int64
Shops_1
5000 non-null int64
Shops_2
5000 non-null bool
Price
5000 non-null float64
dtypes: bool(3), float64(4), int64(13)
memory usage: 678.8 KB
>>> X = df.iloc[:, 0:19]
>>> y = df.iloc[:, 19]
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
>>> sc = StandardScaler()
>>> X_train = sc.fit_transform(X_train)
>>> X_test = sc.transform(X_test)
>>> regressor = RandomForestRegressor(n_estimators=20, random_state=42)
>>> regressor.fit(X_train, y_train)
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
max_depth=None, max_features='auto', max_leaf_nodes=None,
max_samples=None, min_impurity_decrease=0.0,
min_impurity_split=None, min_samples_leaf=1,
min_samples_split=2, min_weight_fraction_leaf=0.0,
n_estimators=20, n_jobs=None, oob_score=False,
random_state=42, verbose=0, warm_start=False)
>>> y_pred = regressor.predict(X_test)
>>>
>>> r2(y_test, y_pred)
0.8068526907254846
>>>

```