Рубежный контроль № 1

Вариант **11**

Задачи

Задача №11: Для набора данных проведите устранение пропусков для одного **(**произвольного**)** категориального признака с использованием метода заполнения отдельной категорией для пропущенных значений**.**

Задача №31: Для набора данных проведите процедуру отбора признаков **(feature selection).** Используйте метод обертывания **(wrapper method),** прямой алгоритм **(sequential forward selection).**

Доп задача

Для пары произвольных колонок данных построить график **"**Диаграмма рассеяния**".**

Датасет

Набор данных по ценам на жилье в России
Описание:

> The aim of this competition is to predict the sale price of each property. The target variable is called price_doc in train.csv.
>
> The training data is from August 2011 to June 2015, and the test set is from July 2015 to May 2016. The dataset also includes information about overall conditions in Russia's economy and finance sector, so you can focus on generating accurate price forecasts for individual properties, without needing to second-guess what the business cycle will do.

In [98]:

```
import pandas as pd
import numpy as np
# Отключим предупреждения
import warnings
warnings.filterwarnings('ignore')
```

In [124]:

```
dataset = pd.read_csv('train.csv', sep=",")
dataset.drop(dataset.iloc[:, 12:291], axis=1, inplace=True)
data = dataset
# Изучим данные
print(data.shape)
print(data.dtypes)
data.head(1000)
```

```
(30471, 13)
id                int64
timestamp        object
full_sq           int64
life_sq         float64
floor           float64
max_floor       float64
material        float64
build_year      float64
num_room        float64
kitch_sq        float64
```

```
state            float64
product_type      object
price_doc          int64
dtype: object
```

Out[124]:

	id	timestamp	full_sq	life_sq	floor	max_floor	material	build_year	num_room	kitch_sq	state	product_type	pric
0	1	2011-08-20	43	27.0	4.0	NaN	NaN	NaN	NaN	NaN	NaN	Investment	58
1	2	2011-08-23	34	19.0	3.0	NaN	NaN	NaN	NaN	NaN	NaN	Investment	60
2	3	2011-08-27	43	29.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	Investment	57
3	4	2011-09-01	89	50.0	9.0	NaN	NaN	NaN	NaN	NaN	NaN	Investment	131
4	5	2011-09-05	77	77.0	4.0	NaN	NaN	NaN	NaN	NaN	NaN	Investment	163
...	...	...	...	...	...	...	...	...	...	...	...	...	
995	996	2012-01-31	67	58.0	10.0	NaN	NaN	NaN	NaN	NaN	NaN	OwnerOccupier	93
996	997	2012-01-31	52	30.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	Investment	41
997	998	2012-01-31	183	183.0	32.0	NaN	NaN	NaN	NaN	NaN	NaN	OwnerOccupier	260
998	999	2012-01-31	155	NaN	11.0	NaN	NaN	NaN	NaN	NaN	NaN	OwnerOccupier	202
999	1000	2012-01-31	54	35.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	Investment	72

1000 rows × 13 columns

In [125]:

```
for col in data.columns:
    pct_missing = np.mean(data[col].isnull())
    if pct_missing != 0:
        print('%s - %s' % (col, round(pct_missing*100)))
```

```
life_sq - 21.0
floor - 1.0
max_floor - 31.0
material - 31.0
build_year - 45.0
num_room - 31.0
kitch_sq - 31.0
state - 44.0
```

Задача №11: Для набора данных проведите устранение пропусков для одного **(**произвольного**)** категориального признака с использованием метода заполнения отдельной категорией для пропущенных значений**.**

Возьмём параметр **max_floor (**этажность здания**),** посчитаем его категориальным признаком и заменим **'NaN'** значения на **'unknown'**

In [126]:

```
print("уникальных значений до замены (max_floor) -", len(data["max_floor"].unique()))
data.loc[data["max_floor"].isnull(),'max_floor'] = 'unknown'
print(">> произвели замену")
print("количество пропусков (max_floor) -", np.mean(data["max_floor"].isnull()))
```

```
print("уникальных значений после замены (max_floor) -", len(data["max_floor"].unique()))
data.head()
```

```
уникальных значений до замены (max_floor) - 50
>> произвели замену
количество пропусков (max_floor) - 0.0
уникальных значений после замены (max_floor) - 50
```

Out[126]:

	id	timestamp	full_sq	life_sq	floor	max_floor	material	build_year	num_room	kitch_sq	state	product_type	price_doc
0	**1**	**2011-08-20**	**43**	**27.0**	**4.0**	**unknown**	**NaN**	**NaN**	**NaN**	**NaN**	**NaN**	**Investment**	**5850000**
1	**2**	**2011-08-23**	**34**	**19.0**	**3.0**	**unknown**	**NaN**	**NaN**	**NaN**	**NaN**	**NaN**	**Investment**	**6000000**
2	**3**	**2011-08-27**	**43**	**29.0**	**2.0**	**unknown**	**NaN**	**NaN**	**NaN**	**NaN**	**NaN**	**Investment**	**5700000**
3	**4**	**2011-09-01**	**89**	**50.0**	**9.0**	**unknown**	**NaN**	**NaN**	**NaN**	**NaN**	**NaN**	**Investment**	**13100000**
4	**5**	**2011-09-05**	**77**	**77.0**	**4.0**	**unknown**	**NaN**	**NaN**	**NaN**	**NaN**	**NaN**	**Investment**	**16331452**

Т.к. количество уникальных не изменилось и видны изменения - произвели замену верно **NaN** на '**unknown**'

Задача №31: Для набора данных проведите процедуру отбора признаков **(feature selection).** Используйте метод обертывания **(wrapper method),** прямой алгоритм **(sequential forward selection).**

In [127]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import LabelEncoder
```

In [128]:

```
for column in data.columns:
    data.loc[data[column].isnull(),column] = 'unknown'
```

In [129]:

```
labelEnc = LabelEncoder()
for column in data.columns:
    if data[column].dtype not in ['float', 'int']:
        data[[column]] = pd.DataFrame(labelEnc.fit_transform(data[column].astype(str)),
columns=[column])
data.head()
```

Out[129]:

	id	timestamp	full_sq	life_sq	floor	max_floor	material	build_year	num_room	kitch_sq	state	product_type	price_d
0	**0**	**0**	**145**	**88**	**32**	**49**	**6**	**119**	**13**	**74**	**5**	**0**	**60**
1	**11109**	**1**	**131**	**74**	**23**	**49**	**6**	**119**	**13**	**74**	**5**	**0**	**62**
2	**22220**	**2**	**145**	**91**	**12**	**49**	**6**	**119**	**13**	**74**	**5**	**0**	**57**
3	**23805**	**3**	**199**	**119**	**40**	**49**	**6**	**119**	**13**	**74**	**5**	**0**	**6**
4	**24916**	**4**	**187**	**149**	**32**	**49**	**6**	**119**	**13**	**74**	**5**	**0**	**10**

In [135]:

```
dataParam = data[list(set(data.columns) - set(["max_floor"]))]
dataTarget = data["max_floor"]
TrainX,TestX,TrainY,TestY = train_test_split(dataParam, dataTarget, test_size=0.3, rando
m_state = 1)
```

In [136]:

```
for column in TrainX.columns:

    randForest = RandomForestRegressor(n_estimators=15)
    randForest.fit(TrainX[[column]], TrainY)
    rFPredict = randForest.predict(TestX[[column]])

    logReg = LogisticRegression()
    logReg.fit(TrainX[[column]], TrainY)
    logRegPredict = logReg.predict(TestX[[column]])

    print("Средняя абсолютная ошибка на колонке ", column)
    print("     RandomForestRegressor",round(mean_absolute_error(TestY, rFPredict),6))
    print("     LogisticRegression",round(mean_absolute_error(TestY, logRegPredict),6))
```

```
Средняя абсолютная ошибка на колонке  build_year
     RandomForestRegressor 12.43684
     LogisticRegression 16.543098
Средняя абсолютная ошибка на колонке  timestamp
     RandomForestRegressor 10.479887
     LogisticRegression 20.054474
Средняя абсолютная ошибка на колонке  full_sq
     RandomForestRegressor 17.039398
     LogisticRegression 19.844235
Средняя абсолютная ошибка на колонке  material
     RandomForestRegressor 8.318574
     LogisticRegression 9.017064
Средняя абсолютная ошибка на колонке  num_room
     RandomForestRegressor 9.992459
     LogisticRegression 8.772151
Средняя абсолютная ошибка на колонке  product_type
     RandomForestRegressor 18.220303
     LogisticRegression 19.844235
Средняя абсолютная ошибка на колонке  life_sq
     RandomForestRegressor 17.316804
     LogisticRegression 19.844235
Средняя абсолютная ошибка на колонке  id
     RandomForestRegressor 14.157289
     LogisticRegression 19.848501
Средняя абсолютная ошибка на колонке  floor
     RandomForestRegressor 17.777255
     LogisticRegression 19.844235
Средняя абсолютная ошибка на колонке  state
     RandomForestRegressor 14.596621
     LogisticRegression 12.112995
Средняя абсолютная ошибка на колонке  kitch_sq
     RandomForestRegressor 6.187306
     LogisticRegression 10.067819
Средняя абсолютная ошибка на колонке  price_doc
     RandomForestRegressor 16.325253
     LogisticRegression 19.844235
```

Доп задание

Для пары произвольных колонок данных построить график "Диаграмма рассеяния".

In [137]:

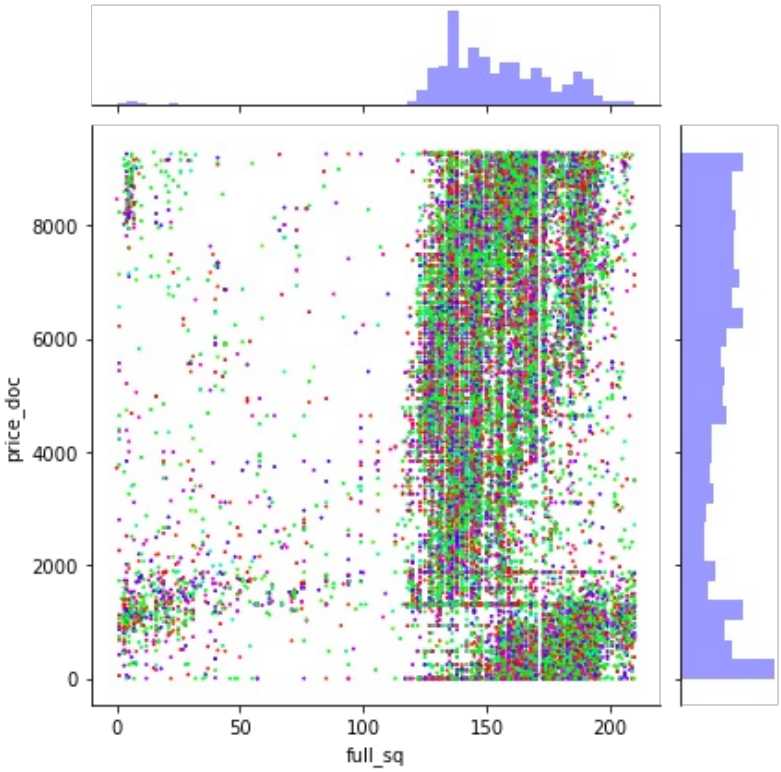
```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

In [138]:

```
colors = np.random.rand(10)
c = plt.cm.gist_rainbow(colors)
sns.jointplot(x=data["full_sq"], y=data["price_doc"], kind='scatter', s=1, color='b', ed
gecolor=c, linewidth=1)
```

Out[138]:

```
<seaborn.axisgrid.JointGrid at 0x1f331c5dee0>
```



In [ ]:

```
```