In [56]:

```
#import os

import numpy as np
import pandas as pd
import csv
import json
import requests as req
import tqdm
import warnings

from pandas.plotting import scatter_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.ticker import FormatStrFormatter
import matplotlib.ticker as ticker

from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
explained_variance_score, mean_absolute_error
from sklearn.preprocessing import (MinMaxScaler,  # Standardization
                                   StandardScaler)

from sklearn.linear_model import LinearRegression

from statistics import median
import scipy
from scipy import stats
import statsmodels.api as sm
import statsmodels.formula.api as smf

warnings.filterwarnings("ignore")
plt.rcParams["font.family"] = "Times New Roman"
my_path = rf"C:\Dmitry"
```

In [57]:

```
df = pd.read_excel(rf"{my_path}\flats_29.03.xlsx")
```

In [58]:

```
df2=df.copy()
df2 = df2[((df2['driving_distance']<20) & (df2['city']!='Санкт-Петербург') & (df2
['JK']!='Южная звезда')) | (df2['city']=='Санкт-Петербург')]
df2.shape
#print(sns.boxplot(x=df2['driving_distance']))
```

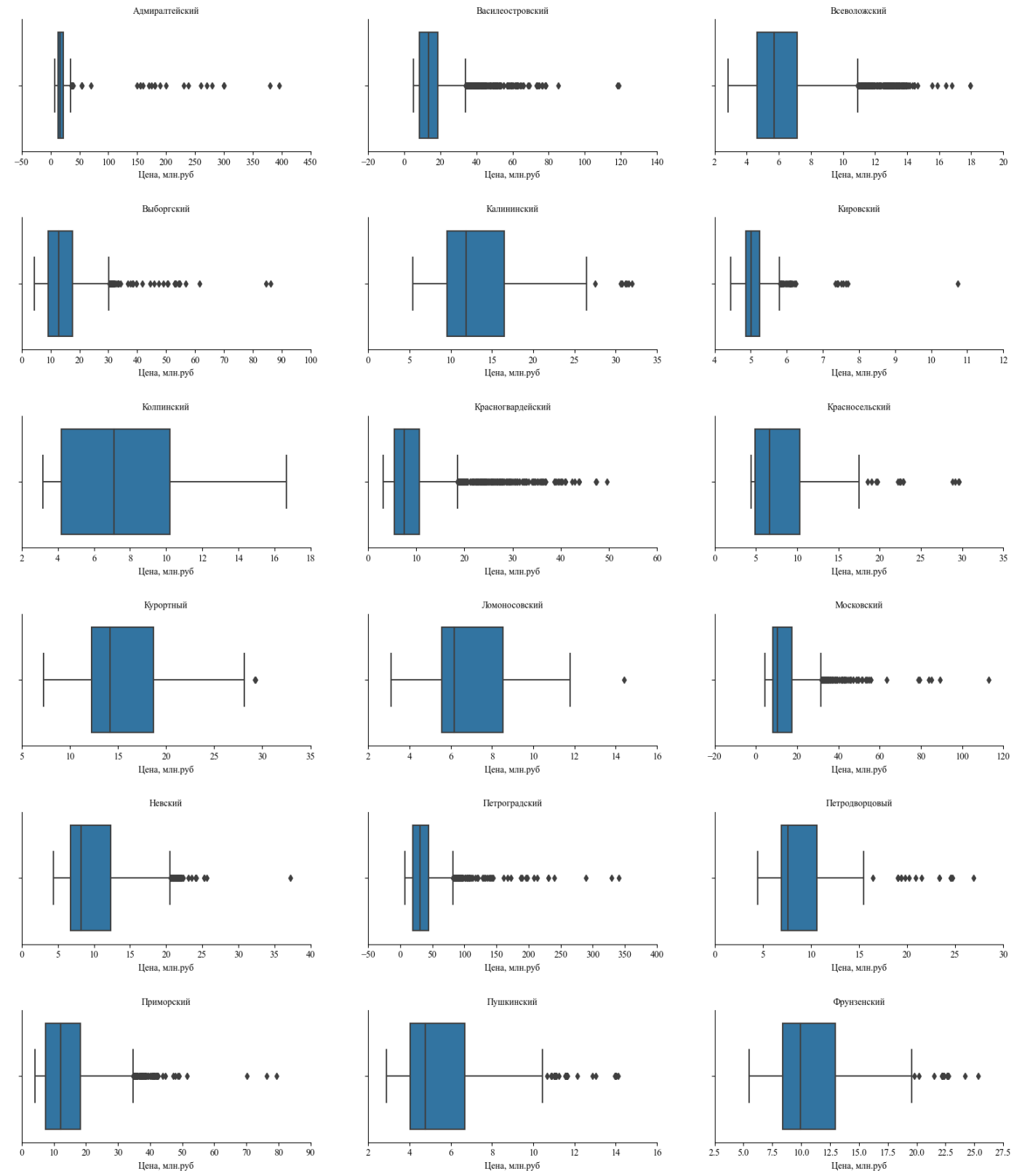
Out[58]:

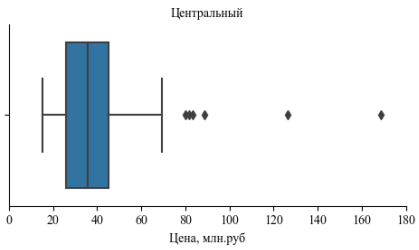
```
(32451, 28)
```

In [59]:

```
#print(sns.histplot(x=df2['full_price']))
df2['full_price']=df2['full_price'].astype(int)
ordered_districts=(list(set(df2.district)))
ordered_districts.sort()
#sns.set_style("plain")
df2['full_price']=df2['full_price']/1000000
```

```
g = sns.FacetGrid(df2, col="district",
                  height=3,
                  aspect=1.7,
                  col_wrap=3,
                  col_order=ordered_districts,
                  sharex=False,
                  margin_titles=True)
g.map(sns.boxplot, "full_price")
g.set_titles(row_template = '{row_name}', col_template = '{col_name}', fontsize=1
0)
g.set(xlabel='Цена, млн.руб')
g.set_xticklabels(style='normal')
g.fig.subplots_adjust(wspace=0.2, hspace=0.5)
df2['full_price']=df2['full_price']*1000000

g.savefig('box_before.pdf', format='pdf', bbox_inches='tight') #, dpi=600
```

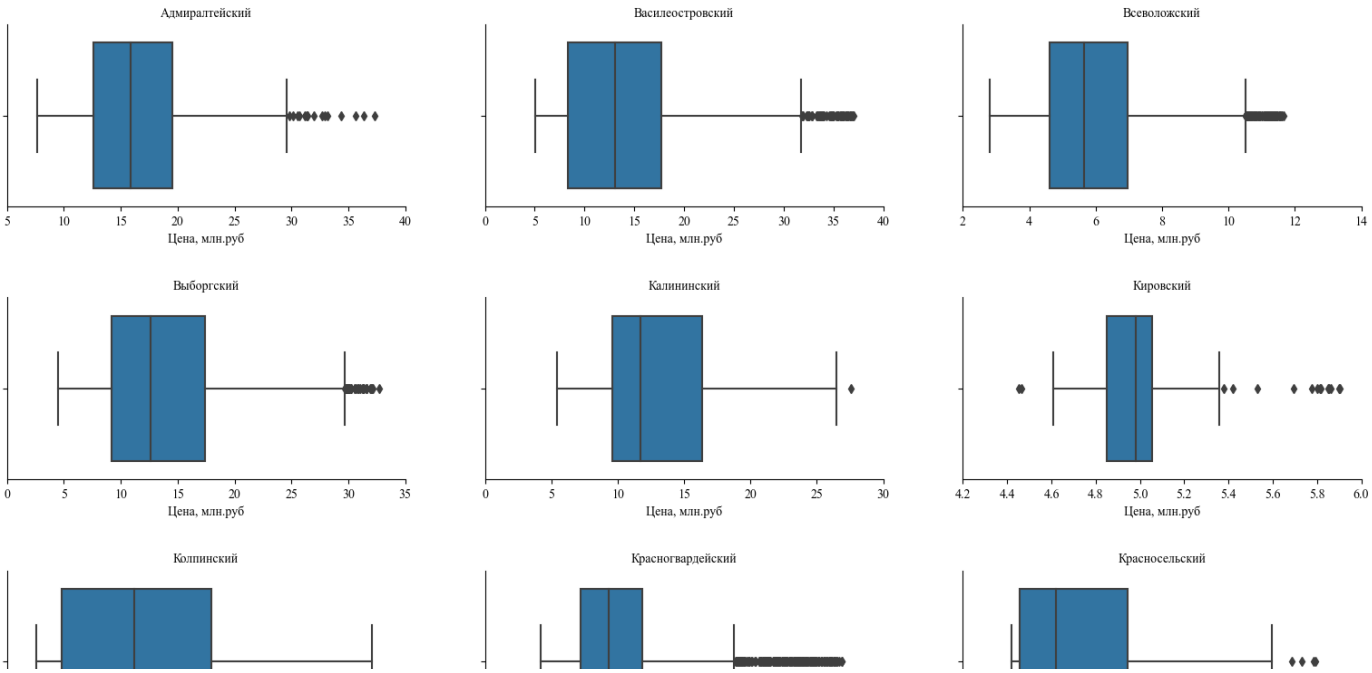


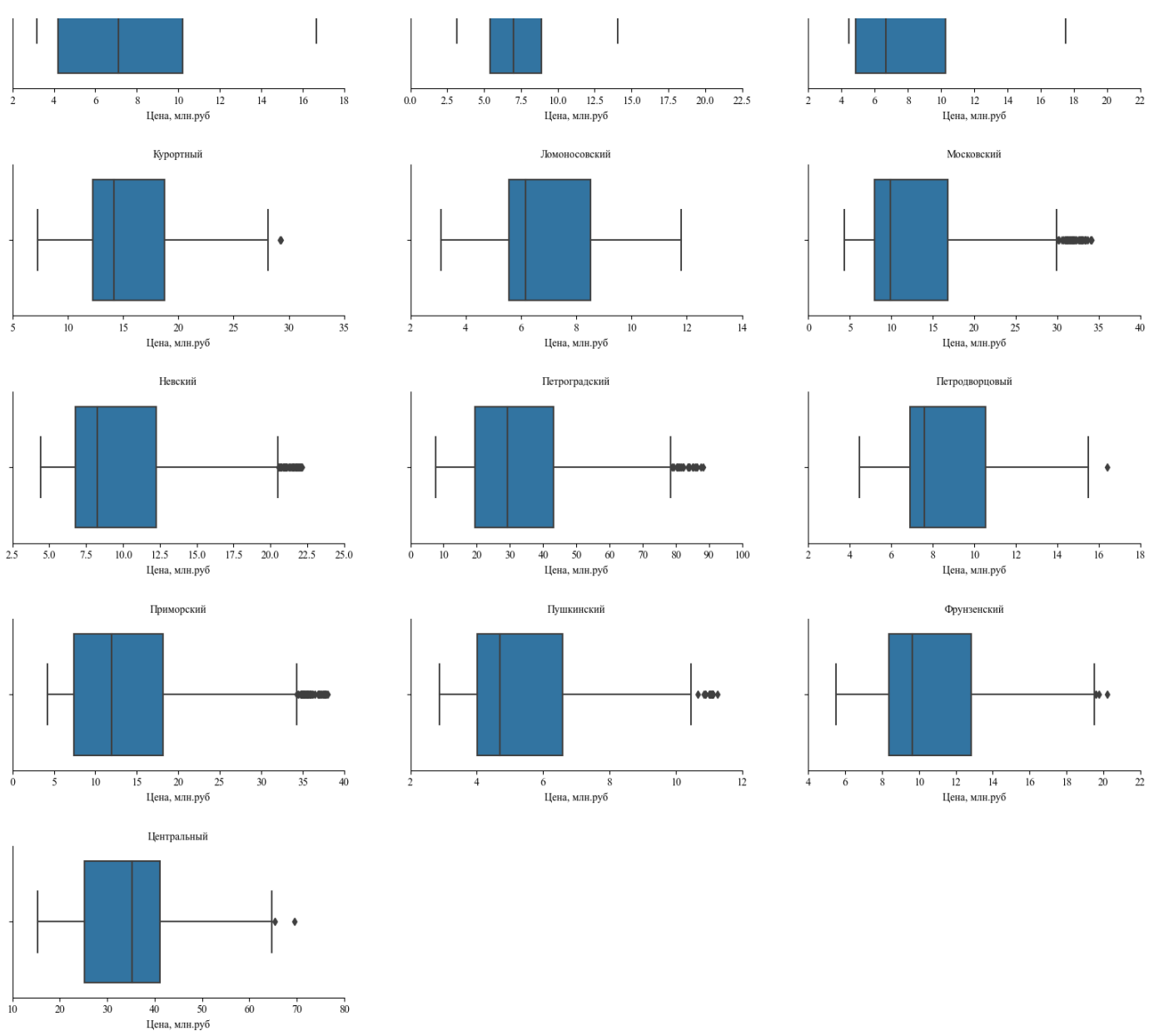


In [60]:

```
df3 = pd.DataFrame(columns = df2.columns)
for i in list(set(df2['district'])):
    q1 = df2[df2['district']==i]['full_price'].quantile(0.25)
    q3 = df2[df2['district']==i]['full_price'].quantile(0.75)
    iqr = q3-q1 #Interquartile range
    #low  = q1-1.5*iqr
    high = q3+1.8*iqr
    df3 = pd.concat([df3, df2.loc[(df2['district'] == i) & (df2['full_price'] < h
igh)]])
#sns.histplot(x=df3['full_price'])
#print(sns.boxplot(x=df2['full_price']))
print(df3.shape)
ordered_districts=(list(set(df3.district)))
ordered_districts.sort()
df3['full_price']=df3['full_price']/1000000
g = sns.FacetGrid(df3, col="district",
                  height=3,
                  aspect=1.7,
                  col_wrap=3,
                  col_order=ordered_districts,
                  sharex=False,
                  margin_titles=True)
g.map(sns.boxplot, "full_price")
g.set_titles(row_template = '{row_name}', col_template = '{col_name}', fontsize=1
0)
g.set(xlabel='Цена, млн.руб')
g.set_xticklabels(style='normal')
g.fig.subplots_adjust(wspace=0.2, hspace=0.5)
df3['full_price']=df3['full_price']*1000000

g.savefig('box_after.pdf', format='pdf', bbox_inches='tight') #, dpi=600
```

```
(31632, 28)
```





In [61]:

```
# for i in list(df3.columns):
#     print(f"Количество NA в {i}: {df3[i].isna().sum()}, доля:
{df3[i].isna().sum()*100/df3.shape[0]:.2f}%")
```

In [62]:

```
df4=df3.copy()

df_jk_height=pd.read_excel(rf"{my_path}\jk_height.xlsx")
df4 = pd.merge(df4, df_jk_height.set_index('JK'), left_on='JK', how='left', right
_index=True)
df4['height_total'] = np.where(df4['height_total_x'].isna(), df4['height_total_y'
], df4['height_total_x'])
df4["height_total"] = df4.groupby("JK")['height_total'].transform(lambda x: x.fil
lna(x.median()))

df4['parking'] = np.where(df4['JK']=='Юттери', 'открытая',
                          np.where(df4['JK']=='Морская набережная. SeaView',
'подземная',
                          np.where(df4['JK']=='Юнтолово', 'открытая',
                          np.where(df4['JK']=='Riverside', 'подземная',
                          np.where(df4['JK']=='Дудергофская линия 3', 'открытая',
df4['parking'])))))

df4 = df4[df4["remont"].notna()]
```

```
df4['house_type']=df4['house_type'].apply(lambda x: x.lower())
df4['remont']=df4['remont'].apply(lambda x: x.lower())
df4['parking']=df4['parking'].apply(lambda x: x.lower())
print(df4.shape)
pd.set_option('display.max_rows', 500)
```

```
(31407, 30)
```

In [63]:

```
date_ready = pd.read_excel(rf"{my_path}\find_date.xlsx")
date_ready['address_d']=(date_ready['JK'].fillna('')+'_'+date_ready['area'].filln
a('')+'_'+date_ready['street'].fillna('')+'_'+date_ready['house'].fillna('')+'_'+
date_ready['building'].fillna(''))
date_ready=date_ready.drop(['JK', 'building', 'area', 'street', 'house'], axis=1
)
df4['address']=(df4['JK'].fillna('')+'_'+df4['area'].fillna('')+'_'+df4['street']
.fillna('')+'_'+df4['house'].fillna('')+'_'+df4['building'].fillna(''))
df4 = pd.merge(df4, date_ready.set_index('address_d'), left_on='address', how='le
ft', right_index=True).reset_index(drop=True)
df4['date_readiness'] = np.where(df4['date_readiness_x'].isna(), df4['date_readin
ess_y'], df4['date_readiness_x'])
```

In [64]:

```
df4['quarts_remained']=np.nan
for i in range(len(df4)):
    if df4['date_readiness'][i]=='2 кв. 2023':
        df4['quarts_remained'][i]=1
    elif df4['date_readiness'][i]=='3 кв. 2023':
        df4['quarts_remained'][i]=2
    elif df4['date_readiness'][i]=='4 кв. 2023':
        df4['quarts_remained'][i]=3
    elif df4['date_readiness'][i]=='1 кв. 2024':
        df4['quarts_remained'][i]=4
    elif df4['date_readiness'][i]=='2 кв. 2024':
        df4['quarts_remained'][i]=5
    elif df4['date_readiness'][i]=='3 кв. 2024':
        df4['quarts_remained'][i]=6
    elif df4['date_readiness'][i]=='4 кв. 2024':
        df4['quarts_remained'][i]=7
    elif df4['date_readiness'][i]=='1 кв. 2025':
        df4['quarts_remained'][i]=8
    elif df4['date_readiness'][i]=='2 кв. 2025':
        df4['quarts_remained'][i]=9
    elif df4['date_readiness'][i]=='3 кв. 2025':
        df4['quarts_remained'][i]=10
    elif df4['date_readiness'][i]=='4 кв. 2025':
        df4['quarts_remained'][i]=11
    elif df4['date_readiness'][i]=='4 кв. 2026':
        df4['quarts_remained'][i]=12
    else:
        df4['quarts_remained'][i]=0

df4=df4.drop(['link', 'building', 'date_readiness', 'date_readiness_x',
'date_readiness_y', 'address', 'area', 'street', 'house', 'floor_house', 'height_
total_x', 'height_total_y'], axis=1)
df4.columns
df4['full_price']=df4['full_price'].astype(int)
df4['floor1']=df4['floor1'].astype(int)
#df4.dtypes
df4.describe().style.format("{:.3f}")
```

Out[64]:

	full_price	full_square	living_square	kitchen	floor1	driving_distance	geo_distance	height_t
count	31407.000	31407.000	31407.000	31407.000	31407.000	31407.000	31407.000	31407.
mean	11011055.488	46.956	21.227	13.399	7.499	5.435	3.498	2.
std	8443757.401	22.271	11.907	6.476	5.039	5.172	3.846	0.
min	2818400.000	17.940	6.800	3.500	1.000	0.208	0.118	2.
25%	5798052.000	31.390	13.200	6.600	3.000	2.030	1.222	2.
50%	8299583.000	39.340	16.060	13.800	7.000	3.842	1.940	2.
75%	13541190.000	59.800	26.900	17.400	11.000	6.262	3.751	2.
max	88239400.000	204.000	120.100	51.000	26.000	53.780	44.716	3.

In [65]:

```
df4.describe().T.style.format("{:.2f}")
median(df4.floor1)
```

Out[65]:

7

In [66]:

```
df4['district'].value_counts()
```

Out[66]:

```
Всеволожский          6822
Приморский            3832
Невский               2886
Василеостровский      2729
Выборгский            2406
Московский            2106
Красногвардейский     2004
Пушкинский            1833
Петроградский         1325
Красносельский         976
Калининский            936
Ломоносовский          920
Колпинский             685
Фрунзенский            531
Петродворцовый         496
Адмиралтейский         371
Курортный              280
Кировский              135
Центральный            134
Name: district, dtype: int64
```

In [67]:

```
districts_reg = ['Адмиралтейский',
  'Василеостровский',
  'Всеволожский',
  'Выборгский',
  'Калининский',
  'Кировский',
  'Колпинский',
  'Красногвардейский',
```

```
  'Красносельский',
  'Курортный',
  'Ломоносовский',
  'Московский',
  'Невский',
  'Петроградский',
  'Петродворцовый',
  'Приморский',
  'Пушкинский',
  'Фрунзенский',
  'Центральный',
#  'Адмиралтейский-Центральный',
#  'Красносельский-Кировский',
#  'Приморский-Курортный'
]
reg_results=pd.DataFrame(columns=['district', 'mae', 'mse', 'rmse', 'mape', 'vrs'
])

list_of_predictions_reg=[]

list_of_coefficients=[]

for cv in range(5):
    reg_results1=pd.DataFrame(columns=['district', 'mae', 'mse', 'rmse', 'mape',
'vrs'])
    reg_results1['district']=districts_reg

    for distr in (districts_reg):
        df5=df4.copy()
        if (distr=='Адмиралтейский-Центральный'):
            df5 = df5[(df5["district"] == 'Адмиралтейский')|(df5["district"] == '
Центральный')]
        elif (distr=='Красносельский-Кировский'):
            df5 = df5[(df5["district"] == 'Красносельский')|(df5["district"] == '
Кировский')]
        elif (distr=='Приморский-Курортный'):
            df5 = df5[(df5["district"] == 'Приморский')|(df5["district"] == 'Куро
ртный')]
        else:
            df5 = df5[df5["district"] == distr]

        df5=df5.drop(['station_name_geo'], axis=1)

        df5Dummies = pd.get_dummies(df5, drop_first=True)

        y = df5Dummies["full_price"]
        X = df5Dummies.drop("full_price", axis=1).values

        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3
)

        scaler = MinMaxScaler()
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)

        reg = LinearRegression()

        reg_model = reg.fit(X_train, y_train)

        predictions = reg_model.predict(X_test)

        while explained_variance_score(y_test,predictions)<0:
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
```

```
0.3)

            scaler = MinMaxScaler()
            X_train = scaler.fit_transform(X_train)
            X_test = scaler.transform(X_test)

            reg = LinearRegression()

            reg_model = reg.fit(X_train, y_train)
            predictions = reg_model.predict(X_test)

            if explained_variance_score(y_test,predictions)>0:
                break

        reg_results1['mae'] = np.where(reg_results1['district']==distr, mean_abso
lute_error(y_test,predictions), reg_results1['mae'])
        reg_results1['mse'] = np.where(reg_results1['district']==distr, mean_squa
red_error(y_test,predictions), reg_results1['mse'])
        reg_results1['rmse'] = np.where(reg_results1['district']==distr, np.sqrt(
mean_squared_error(y_test,predictions)), reg_results1['rmse'])
        reg_results1['mape'] = np.where(reg_results1['district']==distr, mean_abs
olute_percentage_error(y_test,predictions), reg_results1['mape'])
        reg_results1['vrs'] = np.where(reg_results1['district']==distr, explained
_variance_score(y_test,predictions), reg_results1['vrs'])


        prediction=pd.DataFrame({'y_test':y_test.reset_index(drop=True), 'predict
ions':pd.DataFrame(predictions)[0]})
        list_of_predictions_reg.append(prediction)


        model_bench = sm.OLS(y, X)
        res_bench = model_bench.fit(cov_type='HC0') # heteroskedasticity robust s
td. errors
        reg_coef=pd.merge(pd.DataFrame(res_bench.params, columns=['coefficients']
), pd.DataFrame(res_bench.pvalues, columns=['pvalue']), left_index=True,
right_index=True).set_axis(list(df5Dummies.drop("full_price", axis=1).columns),
axis=0)
        reg_coef['abs_coef']=reg_coef['coefficients'].abs()
        reg_coef=reg_coef[reg_coef['pvalue']<0.05].sort_values(by=['abs_coef'], a
scending=True)
        #reg_coef.iloc[:5]
        list_of_coefficients.append(reg_coef)

        del reg
    reg_results=pd.concat([reg_results, reg_results1])
```

In [69]:

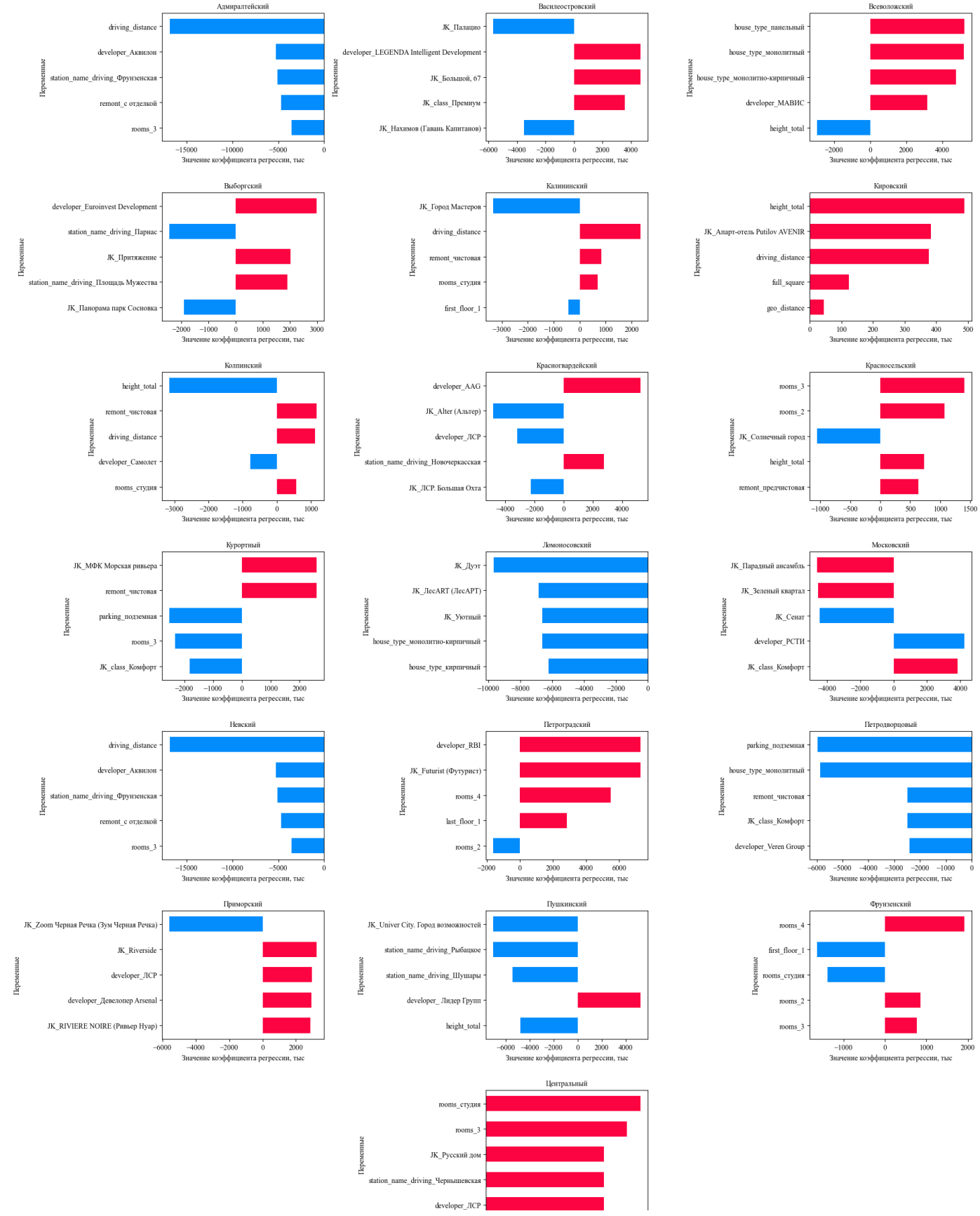
```
pltRowsNmb, pltColsNmb = 7, 3

fig = plt.figure(figsize=[20, 30])

plt.subplots_adjust(wspace=1, hspace=0.4)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 1)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[0].tail
(5).coefficients]
plt.barh(list(list_of_coefficients[0].tail(5).index), list_of_coefficients[0].tai
l(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[0], fontsize = 10)
```

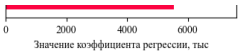
```
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 2)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[1].tail
(5).coefficients]
plt.barh(list(list_of_coefficients[1].tail(5).index), list_of_coefficients[1].tai
l(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[1], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 3)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[2].tail
(5).coefficients]
plt.barh(list(list_of_coefficients[2].tail(5).index), list_of_coefficients[2].tai
l(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[2], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 4)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[3].tail
(5).coefficients]
plt.barh(list(list_of_coefficients[3].tail(5).index), list_of_coefficients[3].tai
l(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[3], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 5)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[4].tail
(5).coefficients]
plt.barh(list(list_of_coefficients[4].tail(5).index), list_of_coefficients[4].tai
l(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[4], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 6)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[5].tail
(5).coefficients]
plt.barh(list(list_of_coefficients[5].tail(5).index), list_of_coefficients[5].tai
l(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[5], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 7)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[6].tail
(5).coefficients]
plt.barh(list(list_of_coefficients[6].tail(5).index), list_of_coefficients[6].tai
l(5).coefficients*(10**-3),
```

```
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[6], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 8)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[7].tail
(5).coefficients]
plt.barh(list(list_of_coefficients[7].tail(5).index), list_of_coefficients[7].tai
l(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[7], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 9)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[8].tail
(5).coefficients]
plt.barh(list(list_of_coefficients[8].tail(5).index), list_of_coefficients[8].tai
l(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[8], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 10)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[9].tail
(5).coefficients]
plt.barh(list(list_of_coefficients[9].tail(5).index), list_of_coefficients[9].tai
l(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[9], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 11)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[10].tai
l(5).coefficients]
plt.barh(list(list_of_coefficients[10].tail(5).index), list_of_coefficients[10].
tail(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[10], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 12)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[11].tai
l(5).coefficients]
plt.barh(list(list_of_coefficients[11].tail(5).index), list_of_coefficients[12].
tail(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[11], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 13)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[0].tail
```

```
(5).coefficients]
plt.barh(list(list_of_coefficients[0].tail(5).index), list_of_coefficients[0].tai
l(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[12], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 14)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[13].tai
l(5).coefficients]
plt.barh(list(list_of_coefficients[13].tail(5).index), list_of_coefficients[13].
tail(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[13], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 15)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[14].tai
l(5).coefficients]
plt.barh(list(list_of_coefficients[14].tail(5).index), list_of_coefficients[14].
tail(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[14], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 16)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[15].tai
l(5).coefficients]
plt.barh(list(list_of_coefficients[15].tail(5).index), list_of_coefficients[15].
tail(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[15], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 17)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[16].tai
l(5).coefficients]
plt.barh(list(list_of_coefficients[16].tail(5).index), list_of_coefficients[16].
tail(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[16], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 18)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[17].tai
l(5).coefficients]
plt.barh(list(list_of_coefficients[17].tail(5).index), list_of_coefficients[17].
tail(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[17], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)
```

```
ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 20)
clrs = ['#038cfc' if (x < 0) else '#fc0341' for x in list_of_coefficients[18].tai
l(5).coefficients]
plt.barh(list(list_of_coefficients[18].tail(5).index), list_of_coefficients[18].
tail(5).coefficients*(10**-3),
         height=0.6,
         color=clrs)
ax.set_title(districts_reg[18], fontsize = 10)
ax.set_xlabel('Значение коэффициента регрессии, тыс', fontsize = 10)
ax.set_ylabel('Переменные', fontsize = 10)

plt.savefig('reg_biggest_vars.pdf', format='pdf', bbox_inches='tight')
```





In [70]:

```
reg_results = reg_results.groupby(["district"])['mae', 'mse', 'rmse', 'mape', 'vrs'].mean().reset_index()
reg_results
reg_results.to_csv('reg_results.csv', index=False)
```

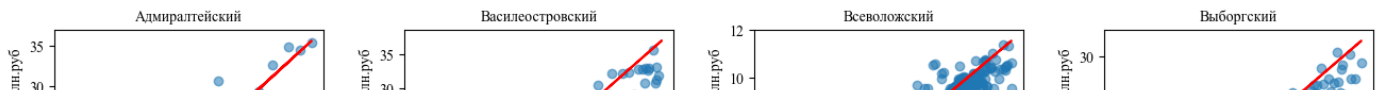
In [71]:

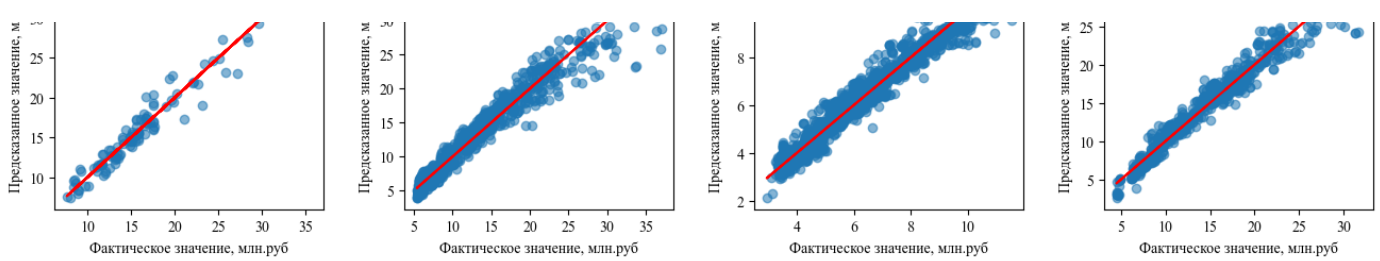
```
pltRowsNmb, pltColsNmb = 5, 4

fig = plt.figure(figsize=[16, 20]) #10, 25

plt.subplots_adjust(wspace=0.3, hspace=0.4) #wspace=0.4, hspace=0.4

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 1)
plt.scatter(x=list_of_predictions_reg[0].y_test/1000000,
y=list_of_predictions_reg[0].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[0].y_test/1000000, list_of_predictions_reg[0].y_test/1000000, 'r')
ax.set_title(districts_reg[0], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(5))
ax.xaxis.set_major_locator(ticker.MultipleLocator(5))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 2)
plt.scatter(x=list_of_predictions_reg[1].y_test/1000000,
y=list_of_predictions_reg[1].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[1].y_test/1000000, list_of_predictions_reg[1].y_test/1000000, 'r')
ax.set_title(districts_reg[1], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(5))
ax.xaxis.set_major_locator(ticker.MultipleLocator(5))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 3)
plt.scatter(x=list_of_predictions_reg[2].y_test/1000000,
y=list_of_predictions_reg[2].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[2].y_test/1000000, list_of_predictions_reg[2].y_test/1000000, 'r')
ax.set_title(districts_reg[2], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(2))
ax.xaxis.set_major_locator(ticker.MultipleLocator(2))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 4)
plt.scatter(x=list_of_predictions_reg[3].y_test/1000000,
y=list_of_predictions_reg[3].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[3].y_test/1000000, list_of_predictions_reg[3].y_test/1000000, 'r')
ax.set_title(districts_reg[3], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
```

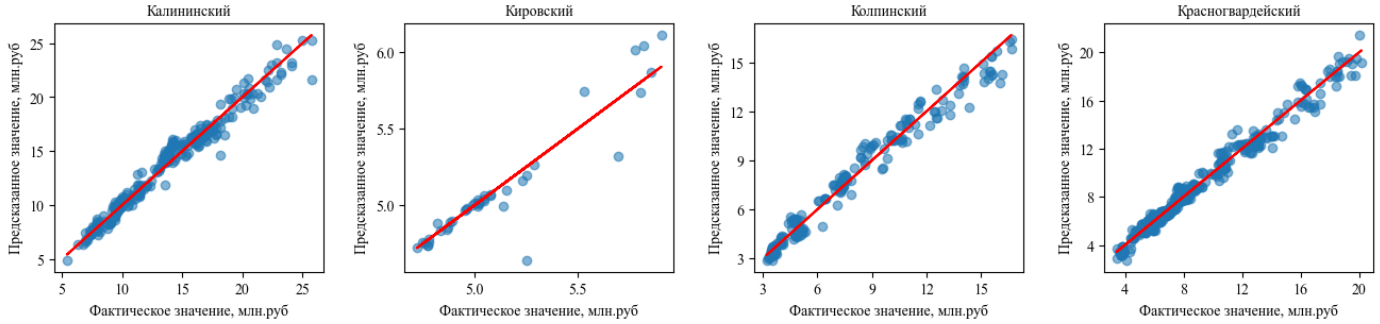
```
ax.yaxis.set_major_locator(ticker.MultipleLocator(5))
ax.xaxis.set_major_locator(ticker.MultipleLocator(5))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 5)
plt.scatter(x=list_of_predictions_reg[4].y_test/1000000,
y=list_of_predictions_reg[4].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[4].y_test/1000000, list_of_predictions_reg[4].y_
test/1000000, 'r')
ax.set_title(districts_reg[4], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(5))
ax.xaxis.set_major_locator(ticker.MultipleLocator(5))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 6)
plt.scatter(x=list_of_predictions_reg[5].y_test/1000000,
y=list_of_predictions_reg[5].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[5].y_test/1000000, list_of_predictions_reg[5].y_
test/1000000, 'r')
ax.set_title(districts_reg[5], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.1f'))
ax.xaxis.set_major_formatter(FormatStrFormatter('%.1f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(0.5))
ax.xaxis.set_major_locator(ticker.MultipleLocator(0.5))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 7)
plt.scatter(x=list_of_predictions_reg[6].y_test/1000000,
y=list_of_predictions_reg[6].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[6].y_test/1000000, list_of_predictions_reg[6].y_
test/1000000, 'r')
ax.set_title(districts_reg[6], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(3))
ax.xaxis.set_major_locator(ticker.MultipleLocator(3))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 8)
plt.scatter(x=list_of_predictions_reg[7].y_test/1000000,
y=list_of_predictions_reg[7].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[7].y_test/1000000, list_of_predictions_reg[7].y_
test/1000000, 'r')
ax.set_title(districts_reg[7], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(4))
ax.xaxis.set_major_locator(ticker.MultipleLocator(4))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 9)
plt.scatter(x=list_of_predictions_reg[8].y_test/1000000,
y=list_of_predictions_reg[8].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[8].y_test/1000000, list_of_predictions_reg[8].y_
test/1000000, 'r')
ax.set_title(districts_reg[8], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(3))
ax.xaxis.set_major_locator(ticker.MultipleLocator(3))
```

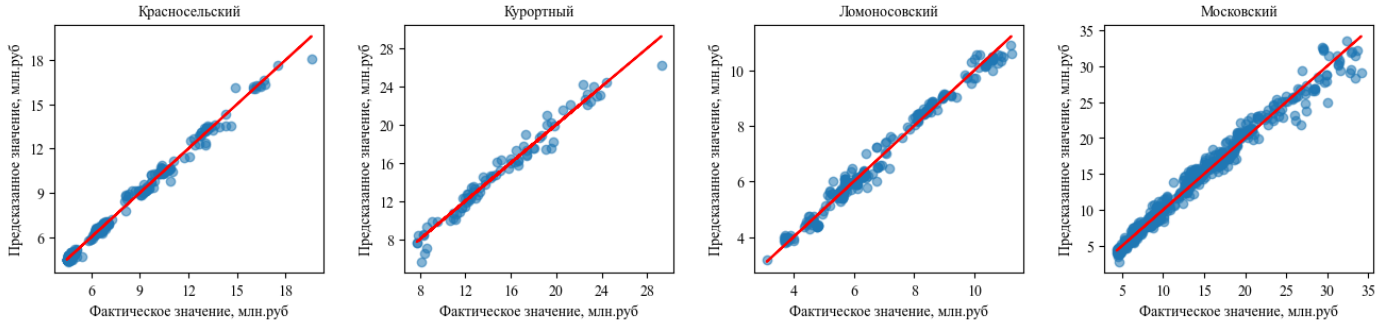
```
ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 10)
plt.scatter(x=list_of_predictions_reg[9].y_test/1000000,
y=list_of_predictions_reg[9].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[9].y_test/1000000, list_of_predictions_reg[9].y_
test/1000000, 'r')
ax.set_title(districts_reg[9], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(4))
ax.xaxis.set_major_locator(ticker.MultipleLocator(4))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 11)
plt.scatter(x=list_of_predictions_reg[10].y_test/1000000,
y=list_of_predictions_reg[10].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[10].y_test/1000000, list_of_predictions_reg[10]
.y_test/1000000, 'r')
ax.set_title(districts_reg[10], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(2))
ax.xaxis.set_major_locator(ticker.MultipleLocator(2))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 12)
plt.scatter(x=list_of_predictions_reg[11].y_test/1000000,
y=list_of_predictions_reg[11].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[11].y_test/1000000, list_of_predictions_reg[11]
.y_test/1000000, 'r')
ax.set_title(districts_reg[11], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(5))
ax.xaxis.set_major_locator(ticker.MultipleLocator(5))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 13)
plt.scatter(x=list_of_predictions_reg[12].y_test/1000000,
y=list_of_predictions_reg[12].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[12].y_test/1000000, list_of_predictions_reg[12]
.y_test/1000000, 'r')
ax.set_title(districts_reg[12], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(4))
ax.xaxis.set_major_locator(ticker.MultipleLocator(4))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 14)
plt.scatter(x=list_of_predictions_reg[13].y_test/1000000,
y=list_of_predictions_reg[13].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[13].y_test/1000000, list_of_predictions_reg[13]
.y_test/1000000, 'r')
ax.set_title(districts_reg[13], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(15))
ax.xaxis.set_major_locator(ticker.MultipleLocator(15))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 15)
plt.scatter(x=list_of_predictions_reg[14].y_test/1000000,
```

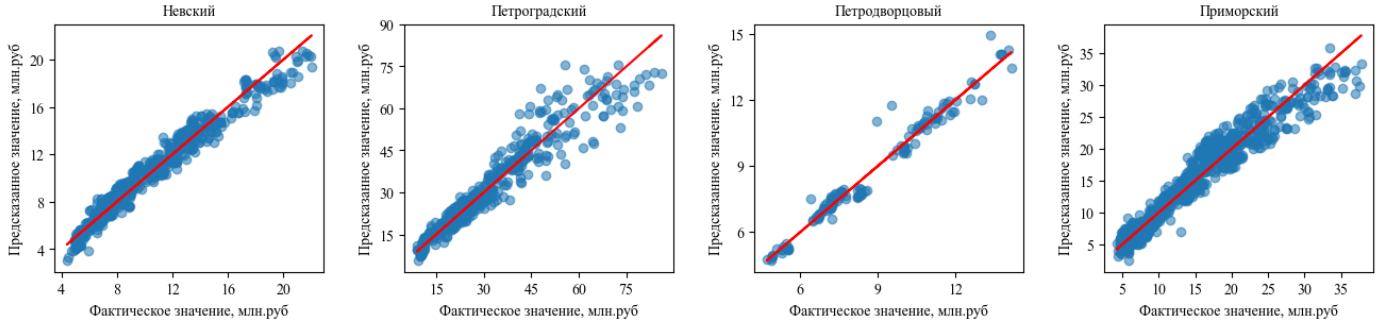
```
y=list_of_predictions_reg[14].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[14].y_test/1000000, list_of_predictions_reg[14]
.y_test/1000000, 'r')
ax.set_title(districts_reg[14], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(3))
ax.xaxis.set_major_locator(ticker.MultipleLocator(3))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 16)
plt.scatter(x=list_of_predictions_reg[15].y_test/1000000,
y=list_of_predictions_reg[15].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[15].y_test/1000000, list_of_predictions_reg[15]
.y_test/1000000, 'r')
ax.set_title(districts_reg[15], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(5))
ax.xaxis.set_major_locator(ticker.MultipleLocator(5))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 17)
plt.scatter(x=list_of_predictions_reg[16].y_test/1000000,
y=list_of_predictions_reg[16].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[16].y_test/1000000, list_of_predictions_reg[16]
.y_test/1000000, 'r')
ax.set_title(districts_reg[16], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(2))
ax.xaxis.set_major_locator(ticker.MultipleLocator(2))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 18)
plt.scatter(x=list_of_predictions_reg[17].y_test/1000000,
y=list_of_predictions_reg[17].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[17].y_test/1000000, list_of_predictions_reg[17]
.y_test/1000000, 'r')
ax.set_title(districts_reg[17], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(2))
ax.xaxis.set_major_locator(ticker.MultipleLocator(2))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 19)
plt.scatter(x=list_of_predictions_reg[18].y_test/1000000,
y=list_of_predictions_reg[18].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[18].y_test/1000000, list_of_predictions_reg[18]
.y_test/1000000, 'r')
ax.set_title(districts_reg[18], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(10))
ax.xaxis.set_major_locator(ticker.MultipleLocator(10))


plt.savefig('reg_results.pdf', format='pdf', bbox_inches='tight') #, dpi=100
```

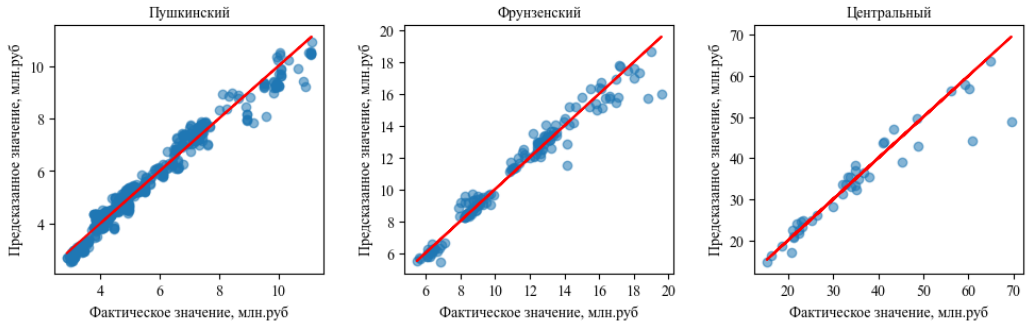












In [72]:

```
#df4.to_csv('df.csv', index=False)
```

In [73]:

```
df4 = pd.read_csv(rf"{my_path}\df_2704.csv")
district_counts=pd.DataFrame(df4['district'].value_counts())
district_counts.reset_index(inplace=True)
district_counts=district_counts.rename(columns={"district": "counts", "index":
"district"})
# district_counts
```

In [74]:

```
reg_results
```

Out[74]:

	district	mae	mse	rmse	mape	vrs
0	Адмиралтейский	957017.85283	1944335934068.83667	1390562.33728	0.05963	0.93947
1	Василеостровский	1150401.37729	2727030314941.29688	1650680.72022	0.08699	0.94321
2	Всеволожский	268395.61114	141880052524.18369	376616.08365	0.04625	0.95772
3	Выборгский	869071.68587	1394314756955.75391	1180193.42985	0.07235	0.95761
4	Калининский	570061.28493	701946345129.61157	833981.88608	0.04149	0.96499
5	Кировский	50055.96764	9972308206.22946	97301.83926	0.00961	0.90082
6	Колпинский	461960.11359	375133231971.74072	612027.19687	0.06287	0.97240
7	Красногвардейский	344740.70731	235446686746.97705	484577.76085	0.04682	0.98426
8	Красносельский	179331.23177	80897541785.40862	283307.55911	0.02210	0.99255
9	Курортный	662035.69024	817312011595.66882	896713.18458	0.04598	0.96349
10	Ломоносовский	188555.62672	70628158598.91739	265156.84068	0.02846	0.98098
11	Московский	679828.08895	1107024443345.68213	1048126.17002	0.05585	0.97558
12	Невский	523170.16658	509415014704.40887	713350.52209	0.05599	0.96788
13	Петроградский	3522314.34192	26010781576391.41406	5093619.23215	0.10817	0.91952
14	Петродворцовый	276221.46765	202286357886.44815	446539.53073	0.03136	0.96538
15	Приморский	1223386.70226	2929746665417.30859	1711197.72263	0.09546	0.94572
16	Пушкинский	262597.00896	109219119104.56493	330303.73855	0.05191	0.96934
17	Фрунзенский	402169.81165	377184677441.41583	607657.96611	0.03660	0.96901
18	Центральный	2249946.52546	12067501743084.71289	3337732.11676	0.06303	0.92514

In [75]:

```
nn_results=pd.read_csv(rf"{my_path}\nn_results.csv")
nn_results=nn_results.drop(['nodes', 'RMSE', 'MAE', 'var_score'], axis=1)
reg_results=pd.read_csv(rf"{my_path}\reg_results.csv")
reg_results1=reg_results.drop(['mae', 'rmse', 'vrs'], axis=1)
all_results = pd.merge(nn_results, reg_results1.set_index('district'), left_on='
district', how='left', right_index=True)
all_results=all_results.rename(columns={"MSE": "MSE_nn", "MAPE": "MAPE_nn", "mse"
: "MSE_reg", "mape": "MAPE_reg"})
all_results['best_model']=np.where(all_results.MSE_nn<all_results.MSE_reg, 'nn',
'reg')
#all_results
```

In [76]:

```
all_results = pd.merge(all_results, district_counts.set_index('district'),
left_on='district', how='left', right_index=True)

all_results
```

Out[76]:

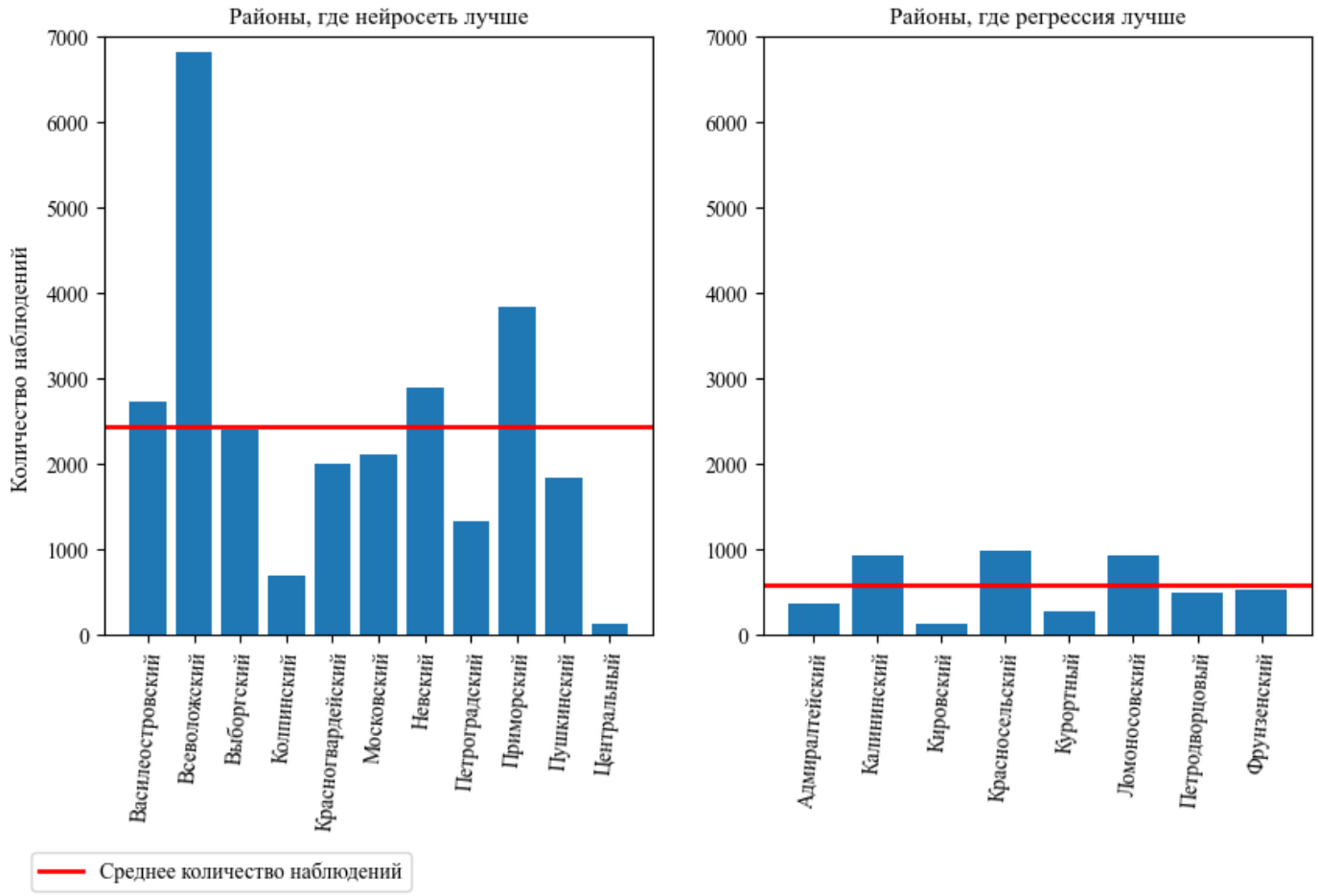
	district	function	optimizer	MSE_nn	MAPE_nn	MSE_reg	MAPE_
0	Адмиралтейский	relu	rmsprop	5735887351510.53027	0.09557	1944335934068.83691	0.05

	district	function	optimizer	MSE_nn	MAPE_nn	MSE_reg	MAPE_
1	Василеостровский	relu	adam	1418911307087.07397	0.04437	2727030314941.29688	0.08
2	Всеволожский	elu	adam	74744417019.71779	0.03286	141880052524.18369	0.04
3	Выборгский	leaky_relu	adam	924909401829.35437	0.04642	1394314756955.75391	0.07
4	Калининский	elu	amsgrad	737171907522.67090	0.04594	701946345129.61157	0.04
5	Кировский	relu	rmsprop	21579066901.88821	0.02074	9972308206.22946	0.00
6	Колпинский	relu	rmsprop	208638616369.57443	0.03863	375133231971.74072	0.06
7	Красногвардейский	elu	adam	150646673048.54709	0.02476	235446686746.97705	0.04
8	Красносельский	relu	rmsprop	125726823392.33476	0.02660	80897541785.40862	0.02
9	Курортный	elu	adam	1011009726210.93762	0.05248	817312011595.66882	0.04
10	Ломоносовский	elu	adam	120010431606.97502	0.03983	70628158598.91739	0.02
11	Московский	elu	adam	683270540556.62439	0.03853	1107024443345.68188	0.05
12	Невский	relu	adam	317077076839.46210	0.03859	509415014704.40887	0.05
13	Петроградский	elu	rmsprop	22396643884614.71094	0.09671	26010781576391.41016	0.10
14	Петродворцовый	elu	rmsprop	365165187408.77631	0.03998	202286357886.44812	0.03
15	Приморский	relu	rmsprop	1332079740650.84595	0.04743	2929746665417.30859	0.09
16	Пушкинский	leaky_relu	amsgrad	51091947384.04011	0.03150	109219119104.56493	0.05
17	Фрунзенский	relu	rmsprop	438916295214.53540	0.03928	377184677441.41583	0.03
18	Центральный	leaky_relu	rmsprop	7859511098218.60156	0.05547	12067501743084.71289	0.06

In [77]:

```
pltRowsNmb, pltColsNmb = 1, 2
csfont = {'fontname':'Times New Roman'}
plt.rcParams["font.family"] = "Times New Roman"

fig = plt.figure(figsize=[10, 5])
plt.subplots_adjust(wspace=0.2, hspace=0.5)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 1)
plt.bar(all_results['district'][all_results['best_model']=='nn'],
all_results['counts'][all_results['best_model']=='nn'])
p1=plt.axhline(all_results['counts'][all_results['best_model']=='nn'].mean(),
color='red', linewidth=2)
p1
ax.set_ylim([0, 7000])
ax.set_xticklabels(all_results['district'][all_results['best_model']=='nn'], rot
ation=85, fontsize=10)
ax.set_title('Районы, где нейросеть лучше', fontsize = 11)
#ax.set_xlabel('Районы', fontsize = 10)
# ax.set_ylabel('MSE, трлн.руб')
# ax.set_xlabel('Количество эпох')
#ax.legend()

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 2)
plt.bar(all_results['district'][all_results['best_model']=='reg'],
all_results['counts'][all_results['best_model']=='reg'])
plt.axhline(all_results['counts'][all_results['best_model']=='reg'].mean(),
color='red', linewidth=2)
ax.set_ylim([0, 7000])
ax.set_xticklabels(all_results['district'][all_results['best_model']=='reg'], ro
tation=85, fontsize=10)
ax.set_title('Районы, где регрессия лучше', fontsize = 11)
```

```
#fig.supxlabel('Район',  fontsize = 11, loc = "bottom")
#fig.text(0.5, -0.3, 'Район',  fontsize = 11, ha='center')
fig.text(0.065, 0.3, 'Количество наблюдений', fontsize = 11, rotation=90)

fig.legend([p1],
    ['Среднее количество наблюдений'],
    loc = 'lower center',
    ncol=1,
    borderaxespad=0.,
    bbox_to_anchor=(0.2, -0.22))

plt.savefig('best_model_districts.pdf', format='pdf', bbox_inches='tight') #,
dpi=300
```



In [89]:

```
print(np.var(all_results['counts'][all_results['best_model']=='nn']))
print(np.var(all_results['counts'][all_results['best_model']=='reg']))
# print(np.var(group1), np.var(group2))

group2=all_results['counts'][all_results['best_model']=='reg']
group1=all_results['counts'][all_results['best_model']=='nn']
# converting the list to array
x = np.array(group1)
y = np.array(group2)

def f_test(group1, group2):
    f = np.var(group1)/np.var(group2)
    nun = x.size-1
    dun = y.size-1
    p_value = 1-scipy.stats.f.cdf(f, nun, dun)
    return f, p_value

# perform F-test
```

```
f_test(x, y)
```

```
2884874.0826446284
92538.984375
```

Out[89]:

```
(31.174689263436477, 7.366024488686396e-05)
```

In [90]:

```
# print(scipy.stats.ttest_ind(all_results['counts']
[all_results['best_model']=='reg'], all_results['counts']
[all_results['best_model']=='nn'], alternative='less'))
print(scipy.stats.ttest_ind(all_results['counts']
[all_results['best_model']=='reg'], all_results['counts']
[all_results['best_model']=='nn'], alternative='less', equal_var=False))
```

```
Ttest_indResult(statistic=-3.3722107923688447, pvalue=0.0031506718397062883)
```

In [25]:

```
# g=sns.scatterplot(data=all_results, x="MAPE_nn", y="MAPE_reg", legend='brief',
size='counts', sizes=(30,300), color='black', alpha=0.8)
# sns.lineplot(data=all_results, x="MAPE_nn", y="MAPE_nn", color='red')
# g.set(ylim=(0, 0.2))
# g.legend(loc='upper right', title='Количество\nнаблюдений\nв районе')
# for i, district in enumerate (all_results.district):
#     plt.annotate(district, (all_results.MAPE_nn[i]+0.7,
all_results.MAPE_reg[i]+0.5) )
```

In [26]:

```
# df6=df4.copy()
# df6=df6.drop(['last_floor', 'first_floor'], axis=1)
# df6=df6.rename({'floor1': 'floor', 'height_total': 'height', 'kitchen': 'kitch
en_square'}, axis=1)
# # df6=df6.rename({}, axis=1)
# # df6=df6.rename({}, axis=1)
# sns.set(style="whitegrid", font_scale=1.3)
# sns.set_style({'font.family':'serif', 'font.serif':['Times New Roman']})
# plt.figure(figsize=(13,13))
# #plt.title('Корреляционная матрица Пирсона',fontsize=25)
#
sns.heatmap(df6.corr(),linewidths=0.1,vmax=1,square=True,cmap="Blues",linecolor=

#             annot=True, annot_kws={"size":12}, cbar_kws={"shrink": .7}, )

# # sns.color_palette("icefire", as_cmap=True)
# plt.savefig('heatmap.pdf', bbox_inches='tight')
```

In [27]:

```
# model =
smf.ols("quarts_remained~full_square+living_square+kitchen+floor+driving_distanc
o_distance+height_total", data=df6).fit()
# model.summary()
```

Для всего города

In [29]:

```
reg_results_city=pd.DataFrame(columns=['mae', 'mse', 'rmse', 'mape', 'vrs'])
```

```
list_of_predictions_reg=[]

list_of_coefficients=[]

for cv in range(5):
    reg_results1=pd.DataFrame(columns=['mae', 'mse', 'rmse', 'mape', 'vrs'])
    # reg_results1['district']=districts_reg
    df5=df4.copy()
    df5=df5.drop(['station_name_geo'], axis=1)

    df5Dummies = pd.get_dummies(df5, drop_first=True)

    y = df5Dummies["full_price"]
    X = df5Dummies.drop("full_price", axis=1).values

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)

    scaler = MinMaxScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    reg = LinearRegression()

    reg_model = reg.fit(X_train, y_train)

    predictions = reg_model.predict(X_test)

    while explained_variance_score(y_test,predictions)<0:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3
)

        scaler = MinMaxScaler()
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)

        reg = LinearRegression()

        reg_model = reg.fit(X_train, y_train)
        predictions = reg_model.predict(X_test)

        if explained_variance_score(y_test,predictions)>0:
            break

    reg_results1.loc[0]=pd.Series({'mae':mean_absolute_error(y_test,predictions),

                    'mse':mean_squared_error(y_test,predictions),
                    'rmse':np.sqrt(mean_squared_error(y_test,predictions)),
                    'mape':mean_absolute_percentage_error(y_test,predictions),
                    'vrs':explained_variance_score(y_test,predictions)})


    prediction=pd.DataFrame({'y_test':y_test.reset_index(drop=True),
'predictions':pd.DataFrame(predictions)[0]})
    list_of_predictions_reg.append(prediction)


    model_bench = sm.OLS(y, X)
    res_bench = model_bench.fit(cov_type='HC0') # heteroskedasticity robust std.
errors
    reg_coef=pd.merge(pd.DataFrame(res_bench.params, columns=['coefficients']), p
d.DataFrame(res_bench.pvalues, columns=['pvalue']), left_index=True, right_index
=True).set_axis(list(df5Dummies.drop("full_price", axis=1).columns), axis=0)
```

```
    reg_coef['abs_coef']=reg_coef['coefficients'].abs()
    reg_coef=reg_coef[reg_coef['pvalue']<0.05].sort_values(by=['abs_coef'], ascen
ding=True)
    #reg_coef.iloc[:5]
    list_of_coefficients.append(reg_coef)

    del reg
    reg_results_city=pd.concat([reg_results_city, reg_results1])
```

In [ ]:

```
reg_results_city
```

Out[ ]:

	mae	mse	rmse	mape	vrs
0	**1.238906e+06**	**4.685955e+12**	**2.164707e+06**	**0.119368**	**0.934952**
0	**1.206481e+06**	**4.416510e+12**	**2.101549e+06**	**0.116842**	**0.935979**
0	**1.191483e+06**	**4.280755e+12**	**2.068998e+06**	**0.115611**	**0.939062**
0	**1.207263e+06**	**4.432755e+12**	**2.105411e+06**	**0.116631**	**0.936937**
0	**1.204768e+06**	**4.370630e+12**	**2.090605e+06**	**0.116244**	**0.940028**

In [44]:

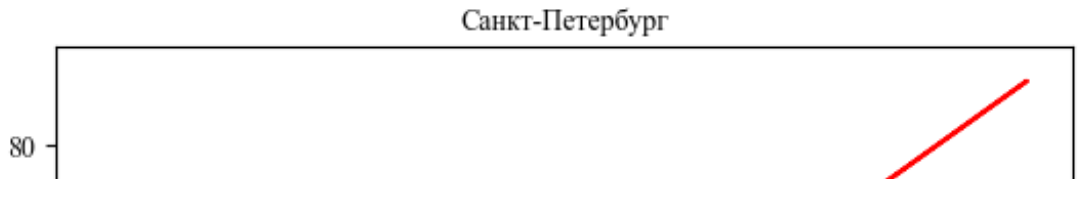
```
pd.set_option('display.float_format', '{:.5f}'.format)
reg_results_city1 = reg_results_city.mean(axis=0).reset_index()
reg_results_city1
# reg_results_city1.to_csv('reg_results_city.csv', index=False)
```

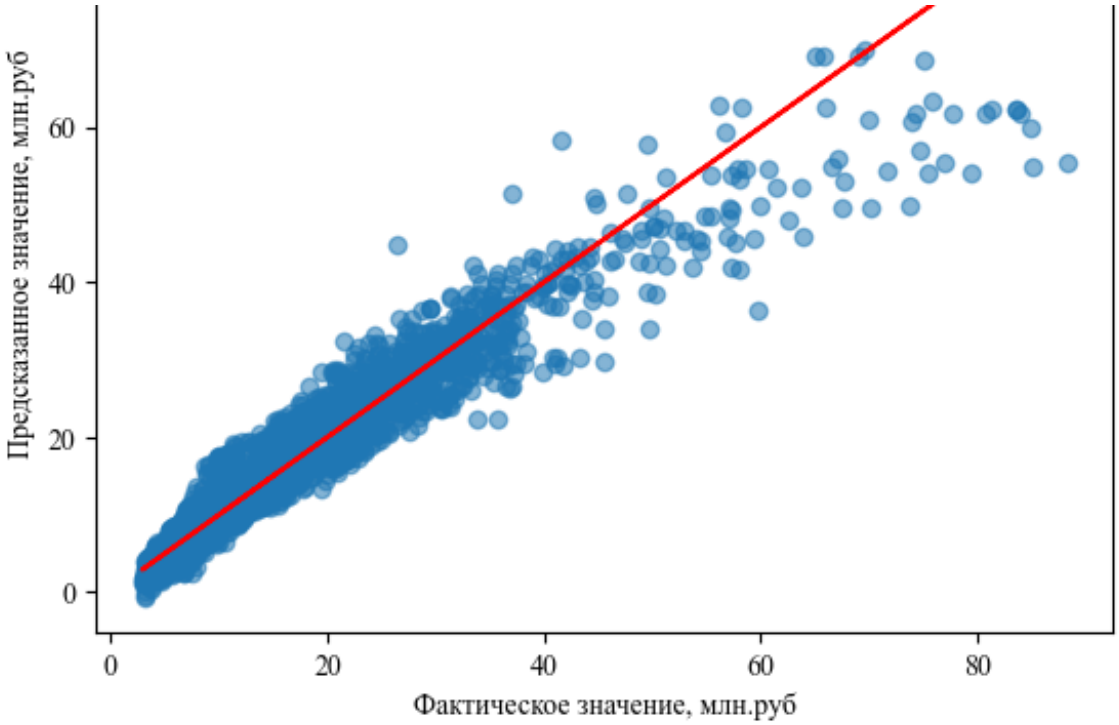
Out[44]:

	index	0
0	**mae**	**1209567.69789**
1	**mse**	**4509788580506.36523**
2	**rmse**	**2123423.19241**
3	**mape**	**0.11729**
4	**vrs**	**0.93667**

In [46]:

```
plt.scatter(x=list_of_predictions_reg[0].y_test/1000000,
y=list_of_predictions_reg[0].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[0].y_test/1000000, list_of_predictions_reg[0].y_
test/1000000, 'r')
plt.title('Санкт-Петербург', fontsize = 10)
plt.xlabel('Фактическое значение, млн.руб', fontsize = 10)
plt.ylabel('Предсказанное значение, млн.руб', fontsize = 10)

plt.savefig('reg_results_city.pdf', format='pdf', bbox_inches='tight') #,
dpi=100
```





Для нейросети

In [48]:

```
from tensorflow import keras
from keras.callbacks import EarlyStopping  # Early Stopping Callback
from keras.layers import Dense, Dropout, Activation
from keras.models import Sequential
from keras.layers import LeakyReLU, PReLU
from keras.wrappers.scikit_learn import KerasRegressor, KerasClassifier

df=df4.copy()
df = pd.get_dummies(df, drop_first=True)

X = df.drop("full_price", axis=1)
y = df["full_price"]
X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.3, random_state=42
    )

scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


model = keras.models.Sequential(
                    [
                        Dense(
                            int(500),
                            activation=keras.layers.ReLU(),
                            kernel_initializer="he_uniform",
                            input_dim=X_train.shape[1],
                        ),
                        Dense(
                            int(300), activation=keras.layers.ReLU(), kernel_init
alizer="he_uniform"
                        ),
                        Dropout(0.3, seed=123),
                        Dense(
                            int(200), activation=keras.layers.ReLU(), kernel_init
```

```
alizer="he_uniform"
                        ),
                        Dropout(0.3, seed=123),
                        Dense(
                            int(100), activation=keras.layers.ReLU(), kernel_init
alizer="he_uniform"
                        ),
                        Dense(1, activation="linear"),
                    ]
                )
model.compile(
                loss="mse",
                optimizer=keras.optimizers.legacy.RMSprop(),
            )

early_stop = EarlyStopping(
                monitor="val_loss", mode="min", verbose=0, patience=5
            )

history=model.fit(
                X_train,
                y_train.values,
                epochs=500,
                batch_size=128,
                validation_data=(X_test, y_test.values),
                callbacks=[early_stop],
                verbose=0,
            )

losses = pd.DataFrame(model.history.history)

    # list_of_losses.append(losses)

predictions_nn = model.predict(X_test, verbose=0)
prediction_nn=pd.DataFrame({'y_test':y_test.reset_index(drop=True),
'predictions':pd.DataFrame(predictions_nn)[0]})
    # list_of_predictions.append(prediction)
```

In [55]:

```
pltRowsNmb, pltColsNmb = 1, 2

fig = plt.figure(figsize=[10, 4]) #10, 25

plt.subplots_adjust(wspace=0.3, hspace=0.4) #wspace=0.4, hspace=0.4

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 1)
plt.scatter(x=prediction_nn.y_test/1000000, y=prediction_nn.predictions/1000000,
alpha=0.55)
plt.plot(prediction_nn.y_test/1000000, prediction_nn.y_test/1000000, 'r')
ax.set_title('Результаты предсказания нейросети', fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 2)
plt.scatter(x=list_of_predictions_reg[0].y_test/1000000,
y=list_of_predictions_reg[0].predictions/1000000, alpha=0.55)
plt.plot(list_of_predictions_reg[0].y_test/1000000, list_of_predictions_reg[0].y_
test/1000000, 'r')
ax.set_title('Результаты предсказания регресии', fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
```

```
plt.savefig('nn_reg_results_city.pdf', format='pdf', bbox_inches='tight')
```