In [24]:

```
#import os
import warnings

import matplotlib.pyplot as plt
from matplotlib.ticker import FormatStrFormatter
import matplotlib.ticker as ticker
import numpy as np
import pandas as pd
import seaborn as sns

from sklearn import preprocessing
from sklearn.ensemble import IsolationForest  # Outlier Detection
from sklearn.metrics import (explained_variance_score, mean_absolute_error,
mean_absolute_percentage_error,
                             mean_squared_error)
from sklearn.model_selection import cross_val_score, train_test_split,
GridSearchCV, RandomizedSearchCV
from sklearn.preprocessing import (MinMaxScaler,  # Standardization
                                   StandardScaler)

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

%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

In [25]:

```
warnings.filterwarnings("ignore")
pd.options.display.max_columns = 60
pd.options.display.max_rows = 60
#plt.rcParams["figure.figsize"] = (7,4)

plt.rcParams["font.family"] = "Times New Roman"

my_path = rf"C:\Dmitry"
```

Получаем данные для районов по отдельности

In [26]:

```
gs = pd.read_csv(rf"{my_path}\grid_search2904_v2_1.csv", delimiter='$')
gs = gs.groupby(["district", "nodes", "function", "optimizer"])['MSE', 'RMSE', 'M
AE', 'MAPE', 'var_score'].mean().reset_index()
best_model=pd.DataFrame(gs.groupby(["district"]).MSE.nsmallest(1).reset_index())
gs=gs.loc[best_model['level_1'],]
gs=gs[gs['district']!='Центральный']
gs

gs_centr = pd.read_csv(rf"{my_path}\grid_search_Centr.csv", delimiter='$')
gs_centr = gs_centr.groupby(["district", "nodes", "function", "optimizer"])
['MSE', 'RMSE', 'MAE', 'MAPE', 'var_score'].mean().reset_index()
```

```
best_model=pd.DataFrame(gs_centr.groupby(["district"]).MSE.nsmallest(1).reset_inc
ex())
best_model
gs_centr=gs_centr.loc[best_model['level_1'],]
gs_centr


best_models=pd.concat([gs, gs_centr]).reset_index(drop=True)
best_models

best_models.to_csv('nn_results.csv', index=False)
```

In [27]:

```
best_models
```

Out[27]:

	district	nodes	function	optimizer	MSE	RMSE	MAE	MAPE	va
0	Адмиралтейский	(900, 600, 300, 200)	relu	rmsprop	5.735887e+12	2.394484e+06	1.529950e+06	0.095571	0.
1	Василеостровский	(700, 600, 300, 200)	relu	adam	1.418911e+12	1.191176e+06	6.972433e+05	0.044369	0.
2	Всеволожский	(500, 300, 200, 100)	elu	adam	7.474442e+10	2.731549e+05	1.937065e+05	0.032861	0.
3	Выборгский	(700, 600, 300, 200)	leaky_relu	adam	9.249094e+11	9.616759e+05	6.535724e+05	0.046423	0.
4	Калининский	(800, 600, 300, 200)	elu	amsgrad	7.371719e+11	8.585849e+05	5.890579e+05	0.045940	0.
5	Кировский	(800, 600, 300, 200)	relu	rmsprop	2.157907e+10	1.464360e+05	1.059646e+05	0.020736	0.
6	Колпинский	(700, 600, 300, 200)	relu	rmsprop	2.086386e+11	4.552913e+05	3.062644e+05	0.038626	0.
7	Красногвардейский	(900, 600, 300, 200)	elu	adam	1.506467e+11	3.881187e+05	2.142705e+05	0.024757	0.
8	Красносельский	(400, 300, 200, 100)	relu	rmsprop	1.257268e+11	3.541551e+05	2.165371e+05	0.026597	0.
9	Курортный	(700, 600, 300, 200)	elu	adam	1.011010e+12	1.005450e+06	7.252853e+05	0.052479	0.
		(700,							

	district	nodes	function	optimizer	MSE	RMSE	MAE	MAPE	va
10	Ломоносовский	500, 300, 200)	elu	adam	1.200104e+12	3.461406e+05	2.629920e+05	0.039854	va
11	Московский	(800, 600, 300, 200)	elu	adam	6.832705e+11	8.265959e+05	5.224075e+05	0.038525	0.
12	Невский	(900, 600, 300, 200)	relu	adam	3.170771e+11	5.630856e+05	3.827311e+05	0.038594	0.
13	Петроградский	(900, 600, 300, 200)	elu	rmsprop	2.239664e+13	4.732492e+06	3.235153e+06	0.096715	0.
14	Петродворцовый	(700, 500, 300, 200)	elu	rmsprop	3.651652e+11	6.042699e+05	3.525507e+05	0.039982	0.
15	Приморский	(900, 600, 300, 200)	relu	rmsprop	1.332080e+12	1.154062e+06	7.140434e+05	0.047433	0.
16	Пушкинский	(900, 600, 300, 200)	leaky_relu	amsgrad	5.109195e+10	2.260228e+05	1.642825e+05	0.031499	0.
17	Фрунзенский	(700, 500, 300, 200)	relu	rmsprop	4.389163e+11	6.621107e+05	4.423768e+05	0.039278	0.
18	Центральный	(700, 600, 300, 200)	leaky_relu	rmsprop	7.859511e+12	2.802338e+06	2.075296e+06	0.055466	0.

In [28]:

```
df_final = pd.read_csv(rf"{my_path}\df_2704.csv")
```

In [31]:

```
list_of_losses=[]
list_of_predictions=[]

for i in range (len(best_models)) :

    df = df_final[df_final["district"] == best_models['district'][[i]].values[0]]

    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)
```

```
    node=list(best_models['nodes'][[i]].astype(str).str.replace("(", "").str.repl
ace(")", "").str.split(", "))[0]

    if best_models['function'][i]=='relu':
        func=keras.layers.ReLU()
    elif best_models['function'][i]=='leaky_relu':
        func=keras.layers.LeakyReLU()
    elif best_models['function'][i]=='elu':
        func=keras.layers.ELU()

    if best_models['optimizer'][i]=='rmsprop':
        opt=keras.optimizers.legacy.RMSprop()
    elif best_models['optimizer'][i]=='adam':
        opt=keras.optimizers.legacy.Adam()
    elif best_models['optimizer'][i]=='amsgrad':
        opt=keras.optimizers.legacy.Adam(amsgrad=True)

    model = keras.models.Sequential(
                    [
                        Dense(
                            int(node[0]),
                            activation=func,
                            kernel_initializer="he_uniform",
                            input_dim=X_train.shape[1],
                        ),
                        Dense(
                            int(node[1]), activation=func, kernel_initializer="he
uniform"
                        ),
                        Dropout(0.3, seed=123),
                        Dense(
                            int(node[2]), activation=func, kernel_initializer="he
uniform"
                        ),
                        Dropout(0.3, seed=123),
                        Dense(
                            int(node[3]), activation=func, kernel_initializer="he
uniform"
                        ),
                        Dense(1, activation="linear"),
                    ]
                )
    model.compile(
                loss="mse",
                optimizer=opt,
            )

    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)
```

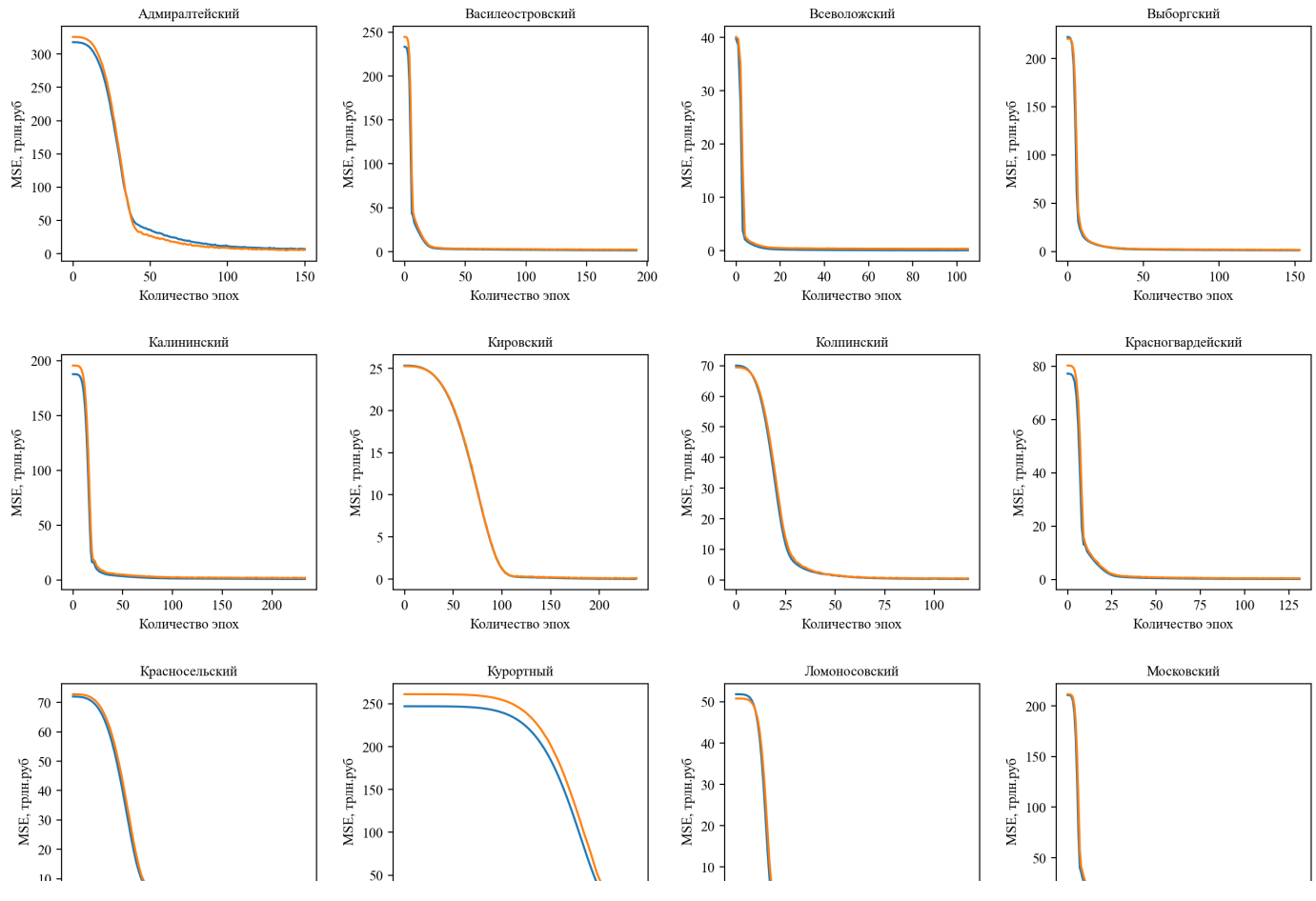
```
    list_of_losses.append(losses)

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

    del model
```

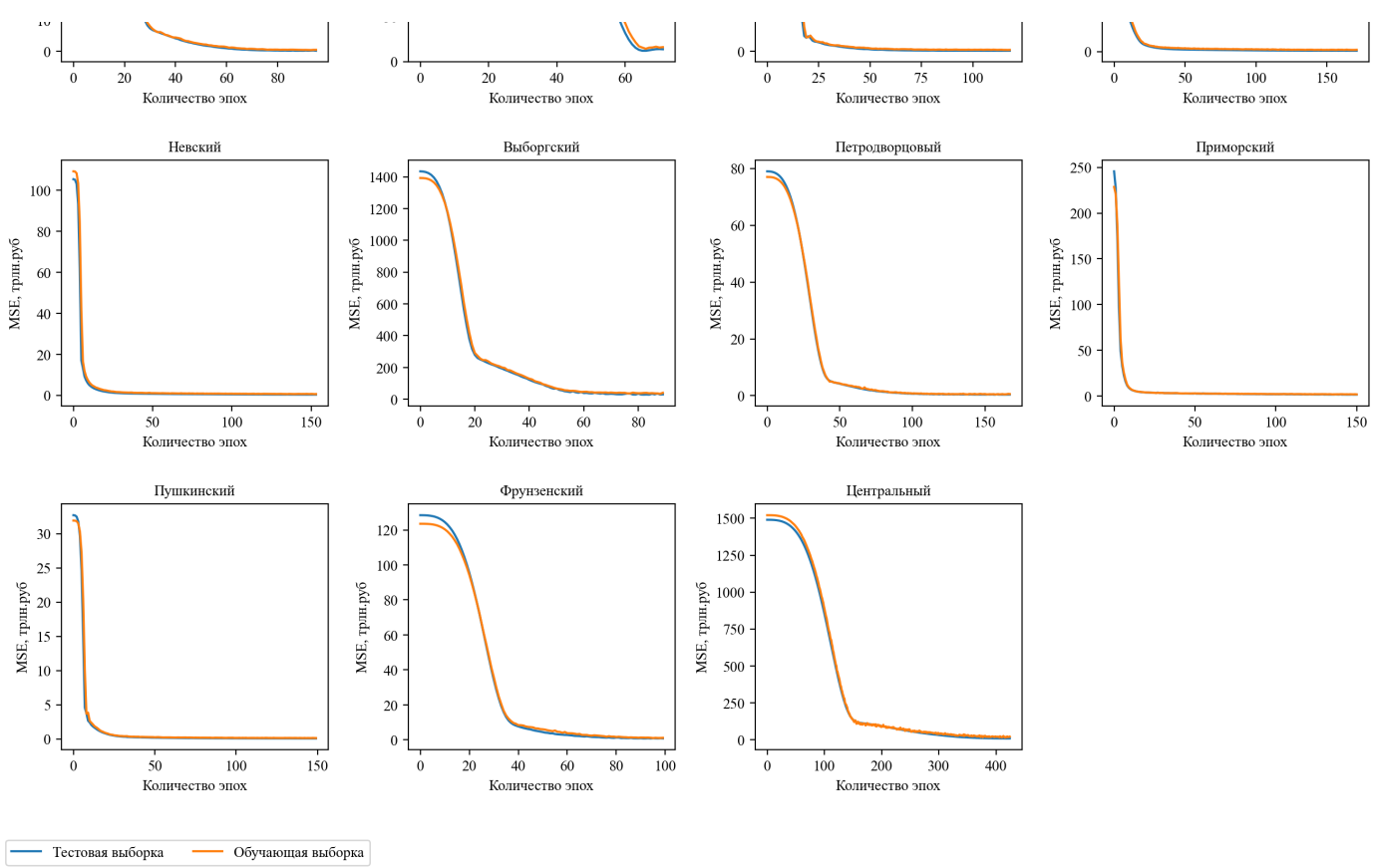
In [32]:

```
pltRowsNmb, pltColsNmb = 5, 4

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

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

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 1)
p1,=plt.plot(list_of_losses[0].val_loss/1000000000000)
p2,=plt.plot(list_of_losses[0].loss/1000000000000)
ax.set_title(best_models['district'][[0]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')
#ax.legend()

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 2)
plt.plot(list_of_losses[1].val_loss/1000000000000)
plt.plot(list_of_losses[1].loss/1000000000000)
ax.set_title(best_models['district'][[1]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 3)
plt.plot(list_of_losses[2].val_loss/1000000000000)
plt.plot(list_of_losses[2].loss/1000000000000)
ax.set_title(best_models['district'][[2]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 4)
plt.plot(list_of_losses[3].val_loss/1000000000000)
plt.plot(list_of_losses[3].loss/1000000000000)
ax.set_title(best_models['district'][[3]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 5)
plt.plot(list_of_losses[4].val_loss/1000000000000)
plt.plot(list_of_losses[4].loss/1000000000000)
ax.set_title(best_models['district'][[4]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 6)
plt.plot(list_of_losses[5].val_loss/1000000000000)
plt.plot(list_of_losses[5].loss/1000000000000)
ax.set_title(best_models['district'][[5]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 7)
plt.plot(list_of_losses[6].val_loss/1000000000000)
```

```
plt.plot(list_of_losses[6].loss/1000000000000)
ax.set_title(best_models['district'][[6]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 8)
plt.plot(list_of_losses[7].val_loss/1000000000000)
plt.plot(list_of_losses[7].loss/1000000000000)
ax.set_title(best_models['district'][[7]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 9)
plt.plot(list_of_losses[8].val_loss/1000000000000)
plt.plot(list_of_losses[8].loss/1000000000000)
ax.set_title(best_models['district'][[8]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 10)
plt.plot(list_of_losses[9].val_loss/1000000000000)
plt.plot(list_of_losses[9].loss/1000000000000)
ax.set_title(best_models['district'][[9]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 11)
plt.plot(list_of_losses[10].val_loss/1000000000000)
plt.plot(list_of_losses[10].loss/1000000000000)
ax.set_title(best_models['district'][[10]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 12)
plt.plot(list_of_losses[11].val_loss/1000000000000)
plt.plot(list_of_losses[11].loss/1000000000000)
ax.set_title(best_models['district'][[11]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 13)
plt.plot(list_of_losses[12].val_loss/1000000000000)
plt.plot(list_of_losses[12].loss/1000000000000)
ax.set_title(best_models['district'][[12]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 14)
plt.plot(list_of_losses[13].val_loss/1000000000000)
plt.plot(list_of_losses[13].loss/1000000000000)
ax.set_title(best_models['district'][[3]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 15)
plt.plot(list_of_losses[14].val_loss/1000000000000)
plt.plot(list_of_losses[14].loss/1000000000000)
ax.set_title(best_models['district'][[14]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 16)
plt.plot(list_of_losses[15].val_loss/1000000000000)
```

```
plt.plot(list_of_losses[15].loss/1000000000000)
ax.set_title(best_models['district'][[15]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 17)
plt.plot(list_of_losses[16].val_loss/1000000000000)
plt.plot(list_of_losses[16].loss/1000000000000)
ax.set_title(best_models['district'][[16]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 18)
plt.plot(list_of_losses[17].val_loss/1000000000000)
plt.plot(list_of_losses[17].loss/1000000000000)
ax.set_title(best_models['district'][[17]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 19)
plt.plot(list_of_losses[18].val_loss/1000000000000)
plt.plot(list_of_losses[18].loss/1000000000000)
ax.set_title(best_models['district'][[18]].values[0], fontsize = 10)
ax.set_ylabel('MSE, трлн.руб')
ax.set_xlabel('Количество эпох')


fig.legend([p1, p2],
    ['Тестовая выборка', 'Обучающая выборка'],
    loc = 'lower center',
    ncol=2,
    borderaxespad=0.,
    bbox_to_anchor=(0.2, 0.055))


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





In [33]:

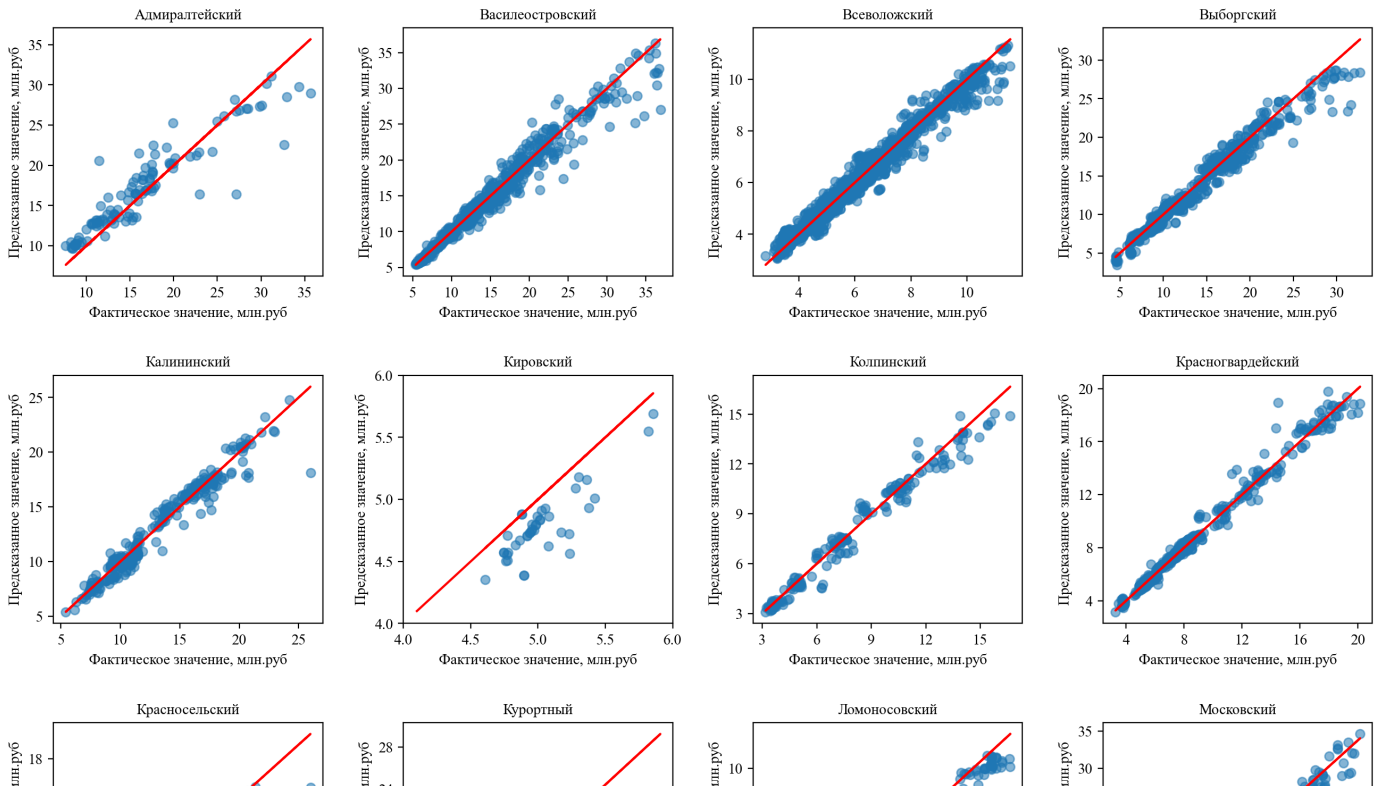
```
pltRowsNmb, pltColsNmb = 5, 4

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

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

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 1)
plt.scatter(x=list_of_predictions[0].y_test/1000000, y=list_of_predictions[0].pre
dictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[0].y_test/1000000, list_of_predictions[0].y_test/100
0000, 'r')
ax.set_title(best_models['district'][[0]].values[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[1].y_test/1000000, y=list_of_predictions[1].pre
dictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[1].y_test/1000000, list_of_predictions[1].y_test/100
0000, 'r')
ax.set_title(best_models['district'][[1]].values[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, 3)
plt.scatter(x=list_of_predictions[2].y_test/1000000, y=list_of_predictions[2].pre
dictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[2].y_test/1000000, list_of_predictions[2].y_test/100
0000, 'r')
```

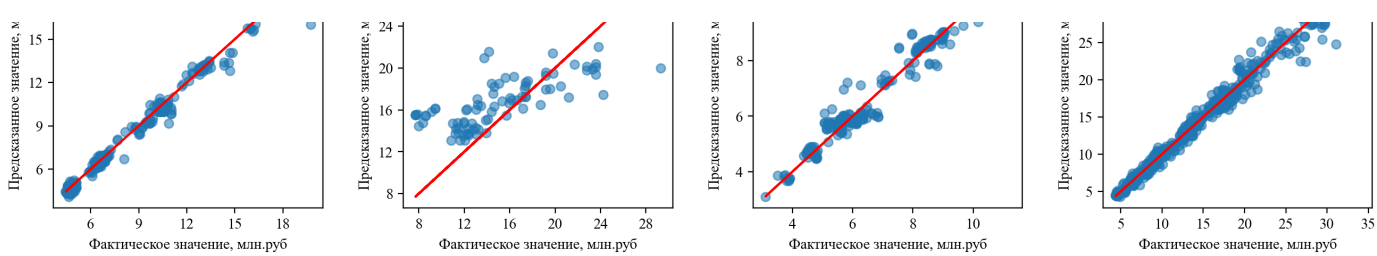
```
ax.set_title(best_models['district'][[2]].values[0], fontsize = 10)
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(2))
ax.xaxis.set_major_locator(ticker.MultipleLocator(2))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 4)
plt.scatter(x=list_of_predictions[3].y_test/1000000, y=list_of_predictions[3].pre
dictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[3].y_test/1000000, list_of_predictions[3].y_test/100
0000, 'r')
ax.set_title(best_models['district'][[3]].values[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, 5)
plt.scatter(x=list_of_predictions[4].y_test/1000000, y=list_of_predictions[4].pre
dictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[4].y_test/1000000, list_of_predictions[4].y_test/100
0000, 'r')
ax.set_title(best_models['district'][[4]].values[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, 6)
df=list_of_predictions[5]
df.loc[len(df.index)] = [4100000, np.nan]
plt.scatter(x=df.y_test/1000000, y=df.predictions/1000000, alpha=0.55)
plt.plot(df.y_test/1000000, df.y_test/1000000, 'r')
plt.ylim(4, 6)
plt.xlim(4, 6)
ax.set_title(best_models['district'][[5]].values[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('%.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[6].y_test/1000000, y=list_of_predictions[6].pre
dictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[6].y_test/1000000, list_of_predictions[6].y_test/100
0000, 'r')
ax.set_title(best_models['district'][[6]].values[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(3))
ax.xaxis.set_major_locator(ticker.MultipleLocator(3))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 8)
plt.scatter(x=list_of_predictions[7].y_test/1000000, y=list_of_predictions[7].pre
dictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[7].y_test/1000000, list_of_predictions[7].y_test/100
0000, 'r')
ax.set_title(best_models['district'][[7]].values[0], fontsize = 10)
```

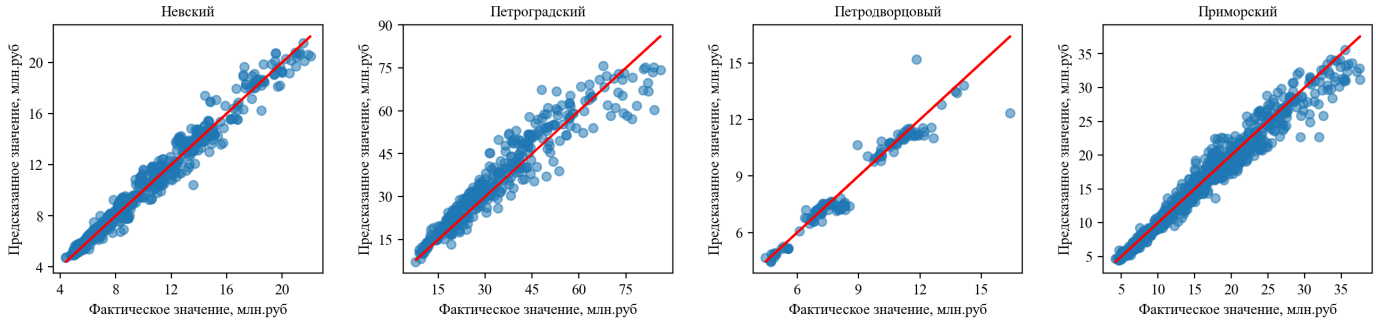
```
ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)
ax.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
ax.yaxis.set_major_locator(ticker.MultipleLocator(4))
ax.xaxis.set_major_locator(ticker.MultipleLocator(4))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 9)
plt.scatter(x=list_of_predictions[8].y_test/1000000, y=list_of_predictions[8].pre
dictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[8].y_test/1000000, list_of_predictions[8].y_test/100
0000, 'r')
ax.set_title(best_models['district'][[8]].values[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(3))
ax.xaxis.set_major_locator(ticker.MultipleLocator(3))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 10)
plt.scatter(x=list_of_predictions[9].y_test/1000000, y=list_of_predictions[9].pre
dictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[9].y_test/1000000, list_of_predictions[9].y_test/100
0000, 'r')
ax.set_title(best_models['district'][[9]].values[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(4))
ax.xaxis.set_major_locator(ticker.MultipleLocator(4))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 11)
plt.scatter(x=list_of_predictions[10].y_test/1000000, y=list_of_predictions[10].p
redictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[10].y_test/1000000, list_of_predictions[10].y_test/1
000000, 'r')
ax.set_title(best_models['district'][[10]].values[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(2))
ax.xaxis.set_major_locator(ticker.MultipleLocator(2))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 12)
plt.scatter(x=list_of_predictions[11].y_test/1000000, y=list_of_predictions[11].p
redictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[11].y_test/1000000, list_of_predictions[11].y_test/1
000000, 'r')
ax.set_title(best_models['district'][[11]].values[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, 13)
plt.scatter(x=list_of_predictions[12].y_test/1000000, y=list_of_predictions[12].p
redictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[12].y_test/1000000, list_of_predictions[12].y_test/1
000000, 'r')
ax.set_title(best_models['district'][[12]].values[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'))
```

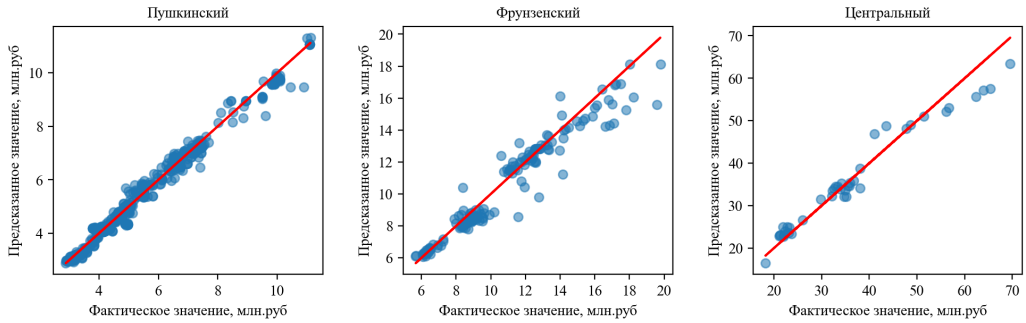
```
ax.yaxis.set_major_locator(ticker.MultipleLocator(4))
ax.xaxis.set_major_locator(ticker.MultipleLocator(4))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 14)
plt.scatter(x=list_of_predictions[13].y_test/1000000, y=list_of_predictions[13].p
redictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[13].y_test/1000000, list_of_predictions[13].y_test/1
000000, 'r')
ax.set_title(best_models['district'][[13]].values[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(15))
ax.xaxis.set_major_locator(ticker.MultipleLocator(15))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 15)
plt.scatter(x=list_of_predictions[14].y_test/1000000, y=list_of_predictions[14].p
redictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[14].y_test/1000000, list_of_predictions[14].y_test/1
000000, 'r')
ax.set_title(best_models['district'][[14]].values[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(3))
ax.xaxis.set_major_locator(ticker.MultipleLocator(3))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 16)
plt.scatter(x=list_of_predictions[15].y_test/1000000, y=list_of_predictions[15].p
redictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[15].y_test/1000000, list_of_predictions[15].y_test/1
000000, 'r')
ax.set_title(best_models['district'][[15]].values[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, 17)
plt.scatter(x=list_of_predictions[16].y_test/1000000, y=list_of_predictions[16].p
redictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[16].y_test/1000000, list_of_predictions[16].y_test/1
000000, 'r')
ax.set_title(best_models['district'][[16]].values[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(2))
ax.xaxis.set_major_locator(ticker.MultipleLocator(2))

ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 18)
plt.scatter(x=list_of_predictions[17].y_test/1000000, y=list_of_predictions[17].p
redictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[17].y_test/1000000, list_of_predictions[17].y_test/1
000000, 'r')
ax.set_title(best_models['district'][[17]].values[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(2))
ax.xaxis.set_major_locator(ticker.MultipleLocator(2))
```

```
ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 19)
plt.scatter(x=list_of_predictions[18].y_test/1000000, y=list_of_predictions[18].p
redictions/1000000, alpha=0.55)
plt.plot(list_of_predictions[18].y_test/1000000, list_of_predictions[18].y_test/1
000000, 'r')
ax.set_title(best_models['district'][[18]].values[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(10))
ax.xaxis.set_major_locator(ticker.MultipleLocator(10))

# ax = fig.add_subplot(pltRowsNmb, pltColsNmb, 20)
# plt.scatter(x=list_of_predictions[19].y_test/1000000,
y=list_of_predictions[19].predictions/1000000)
# plt.plot(list_of_predictions[19].y_test/1000000,
list_of_predictions[19].y_test/1000000, 'r')
# ax.set_title(best_models['district'][[19]].values[0], 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, 22)
# plt.scatter(x=list_of_predictions[20].y_test/1000000,
y=list_of_predictions[20].predictions/1000000)
# plt.plot(list_of_predictions[20].y_test/1000000,
list_of_predictions[20].y_test/1000000, 'r')
# ax.set_title(best_models['district'][[20]].values[0], 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, 23)
# plt.scatter(x=list_of_predictions[21].y_test/1000000,
y=list_of_predictions[21].predictions/1000000)
# plt.plot(list_of_predictions[21].y_test/1000000,
list_of_predictions[21].y_test/1000000, 'r')
# ax.set_title(best_models['district'][[21]].values[0], fontsize = 10)
# ax.set_xlabel('Фактическое значение, млн.руб', fontsize = 10)
# ax.set_ylabel('Предсказанное значение, млн.руб', fontsize = 10)

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









Получаем данные для всего города в целом

In [ ]:

```
gs_f = pd.read_csv(rf"{my_path}\grid_search1705.csv", delimiter='$')
gs_f = gs_f.groupby(["nodes", "function", "optimizer"])['MSE', 'RMSE', 'MAE', 'MA
PE', 'var_score'].mean().reset_index()
best_model_f=pd.DataFrame(gs_f.MSE.nsmallest(1).reset_index())
gs_f=gs_f.loc[best_model_f['index']].reset_index(drop=True)
gs_f
```

Out[ ]:

	nodes	function	optimizer	MSE	RMSE	MAE	MAPE	var_score
0	(500, 300, 200, 100)	relu	rmsprop	1.611866e+12	1.269541e+06	537735.638394	0.040361	0.97817

In [ ]:

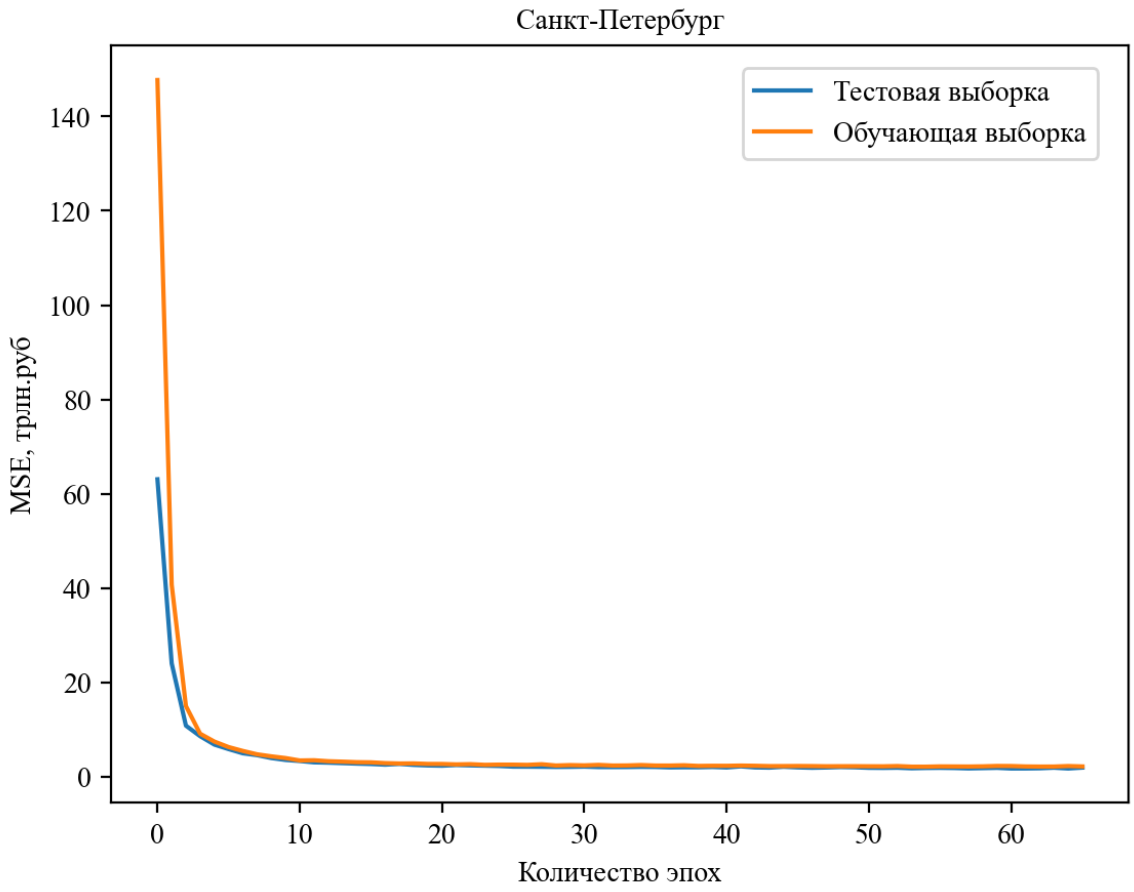
```
df = pd.get_dummies(df_final, drop_first=True)

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)


predictions = model.predict(X_test, verbose=0)
prediction=pd.DataFrame({'y_test':y_test.reset_index(drop=True), 'predictions':p
d.DataFrame(predictions)[0]})
```

In [ ]:

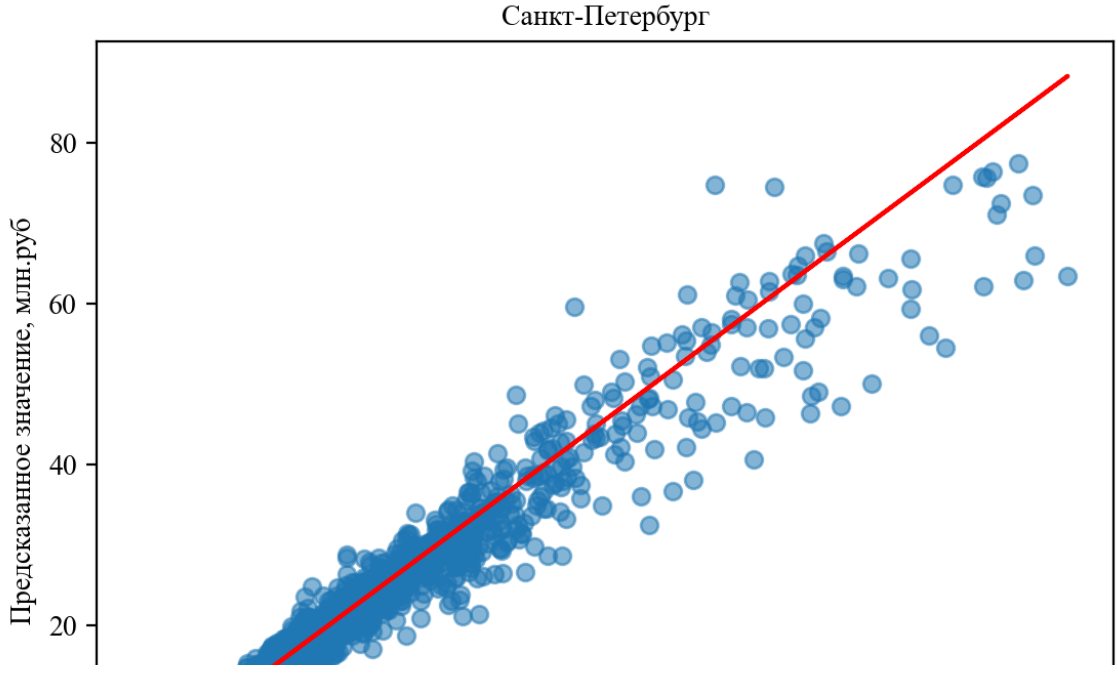
```
p1,=plt.plot(losses.val_loss/1000000000000)
p2,=plt.plot(losses.loss/1000000000000)
plt.title('Санкт-Петербург', fontsize = 10)
plt.ylabel('MSE, трлн.руб')
plt.xlabel('Количество эпох')

plt.legend([p1, p2],
    ['Тестовая выборка', 'Обучающая выборка'],
    loc = 'upper right',
    # ncol=2,
    borderaxespad=0.,
```

```
    bbox_to_anchor=(0.97, 0.97)
    )
plt.savefig('nn_errors_city.pdf', format='pdf', bbox_inches='tight') #, dpi=100
```



In [ ]:

```
plt.scatter(x=prediction.y_test/1000000, y=prediction.predictions/1000000, alpha=
0.55)
plt.plot(prediction.y_test/1000000, prediction.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.yaxis.set_major_formatter(FormatStrFormatter('%.0f'))
# plt.axis.set_major_locator(ticker.MultipleLocator(10))
# plt.xaxis.set_major_locator(ticker.MultipleLocator(10))
plt.savefig('nn_results_city.pdf', format='pdf', bbox_inches='tight') #, dpi=100
```




0
0
20
40
60
80
Фактическое значение, млн.руб