Рубежный контроль №1

Выполнил: Пакало Александр Сергеевич, студент ИУ5-22М

Вариант 13, согласно ему номера задач: 13 и 33 для первой и второй соответственно. Для моей группы доп. требование: для произвольной колонки данных построить гистограмму

Подготовка библиотек и данных

In [1]:
```
# Loading extension for reloading editable packages (pip install -e .)
%load_ext autoreload
```

In [2]:
```
# Reloading editable packages.
%autoreload
# from lab1.main import get_results

import bokeh  # noqa
```

In [3]:
```
import datashader as ds  # noqa
import holoviews as hv  # noqa
import panel as pn  # noqa
from packaging.version import Version  # noqa

min_versions = dict(ds="0.15.1", bokeh="3.2.0", hv="1.16.2", pn="1.2.0")

for lib, ver in min_versions.items():
    v = globals()[lib].__version__
    if Version(v) < Version(ver):
        print("Error: expected {}={}, got {}".format(lib, ver, v))
```

In [4]:
```
hv.extension("bokeh", "matplotlib")
```

In [130…
```
import pathlib  # noqa

try:
    import pandas as pd

    columns = ["depth", "id", "latitude", "longitude", "mag", "place", "time", "type
    path = pathlib.Path("../../data/earthquakes/earthquakes-projected.parq")
    df = pd.read_parquet(path, columns=columns, engine="fastparquet")
    df.head()
except RuntimeError as e:
    print("The data cannot be read: %s" % e)
```

Небольшой разведочный анализ данных

In [7]:
```
print(df.shape)
df.head()
```

```
(696716, 8)
```

Out[7]:

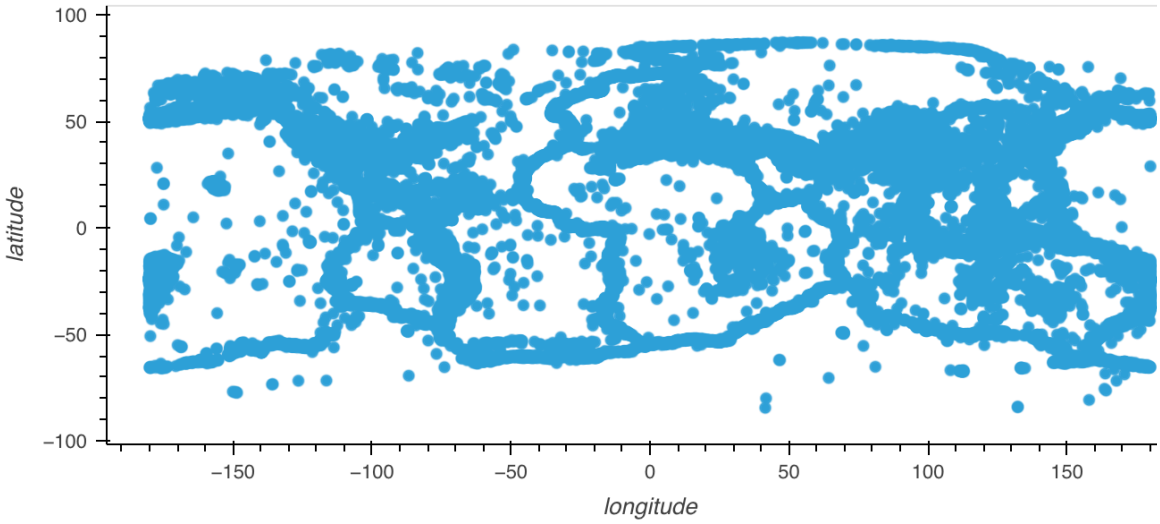
	depth	id	latitude	longitude	mag	place	time	type
0	33.000	usp0009mwt	10.693	-61.162	2.10	12 km NNW of Sangre Grande, Trinidad and Tobago	2000-01-31T23:28:38.420Z	earthquake
1	0.000	rusms00001693	40.212	-107.786	2.10	22 km NNE of Meeker, Colorado	2000-01-31T23:13:15.860Z	explosion
2	33.000	usp0009mws	-1.203	-80.716	4.50	18 km SSW of Montecristi, Ecuador	2000-01-31T23:05:22.010Z	earthquake
3	0.000	rusms00001692	32.742	-109.097	2.90	2 km NNE of Duncan, Arizona	2000-01-31T22:55:53.680Z	explosion
4	4.877	ci9137214	34.375	-119.546	2.21	4km SW of Carpinteria, California	2000-01-31T22:33:54.550Z	earthquake

In [13]:
```
%matplotlib inline
```

In [15]:
```
import hvplot.pandas  # noqa

df.hvplot.scatter(x="longitude", y="latitude")
```

Out[15]:



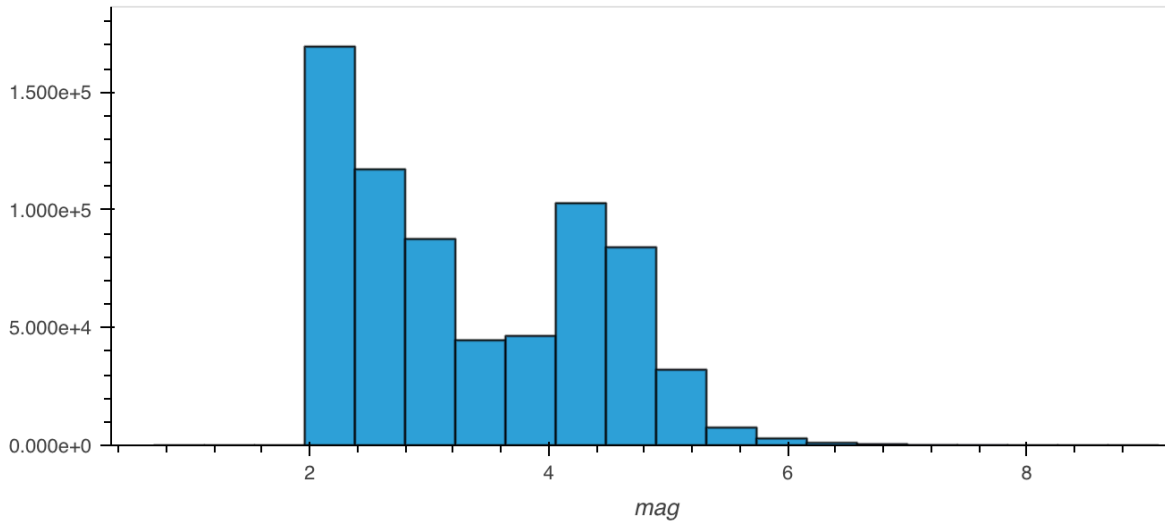
Дополнительное требование для группы

Для произвольной колонки данных построить гистограмму.

Построим гистограмму для колонки "mag" (magnitude - магнитуда).

In [90]:
```
df["mag"].hvplot.hist()
```

Out[90]:



Задача №1 (13)

Для набора данных проведите нормализацию для одного (произвольного) числового признака с использованием функции "обратная зависимость - 1 / X".

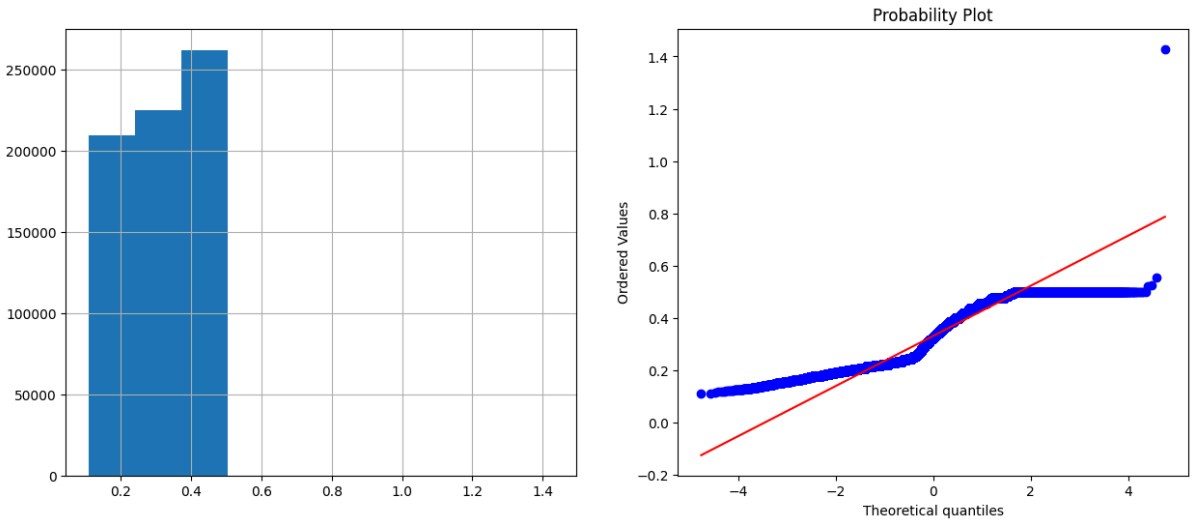
Проведём нормализацию для колонки "mag"

In [91]:
```
import matplotlib.pyplot as plt  # noqa
import numpy as np  # noqa
import pandas as pd  # noqa
import scipy.stats as stats  # noqa
```

In [92]:
```
def diagnostic_plots(df: pd.DataFrame, variable: str):
    plt.figure(figsize=(15, 6))
    # гистограмма
    plt.subplot(1, 2, 1)
    df[variable].hist()
    # Q-Q plot
    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    plt.show()
```

In [132…
```
df["mag_reciprocal"] = 1 / (df["mag"])
```

In [94]:
```
diagnostic_plots(df, "mag_reciprocal")
```



Вывод

Как видно, нормализация такой функцией неудачна.

Задача №2 (33)

Для набора данных проведите процедуру отбора признаков (feature selection). Используйте метод обертывания (wrapper method), алгоритм полного перебора (exhaustive feature selection).

Составим модель регрессии для реализации метода обертывания

Метод обёртывания использует результаты от обучения для выбора лучших признаков. Поставим перед собой цель предсказать магнитуду. Для этого составим модель регрессии на основе `RandomForestRegressor`.

Оставим только землетрясения.

In [141…
```
df["type"].unique()
```

Out[141…
```
array(['earthquake', 'explosion', 'quarry blast', 'other event',
       'sonic boom', 'experimental explosion', 'rock burst', 'ice quake',
       'chemical explosion', 'nuclear explosion', 'mine collapse',
       'Rock Slide', 'mining explosion', 'landslide', 'quarry',
       'acoustic noise', 'not reported', 'collapse',
       'induced or triggered event', 'volcanic eruption'], dtype=object)
```

In [144…
```
df.loc[df["type"] == "earthquake"]
df.head()
```

Out[144…

	depth	id	latitude	longitude	mag	place	time	type
0	33.000	usp0009mwt	10.693	-61.162	2.10	12 km NNW of Sangre Grande, Trinidad and Tobago	2000-01-31T23:28:38.420Z	earthquake
1	0.000	rusms00001693	40.212	-107.786	2.10	22 km NNE of Meeker, Colorado	2000-01-31T23:13:15.860Z	explosion
2	33.000	usp0009mws	-1.203	-80.716	4.50	18 km SSW of Montecristi, Ecuador	2000-01-31T23:05:22.010Z	earthquake
3	0.000	rusms00001692	32.742	-109.097	2.90	2 km NNE of Duncan, Arizona	2000-01-31T22:55:53.680Z	explosion
4	4.877	ci9137214	34.375	-119.546	2.21	4km SW of Carpinteria, California	2000-01-31T22:33:54.550Z	earthquake

Так как методы обёртывания довольно затратны с точки зрения времени вычислений, и алгоритм полного перебора является очень жадным, рассмотрим лишь подмножество данных исходного набора.

In [154…
```
small_df = df.sample(frac=0.002)
print(small_df.shape)
small_df.head()
```

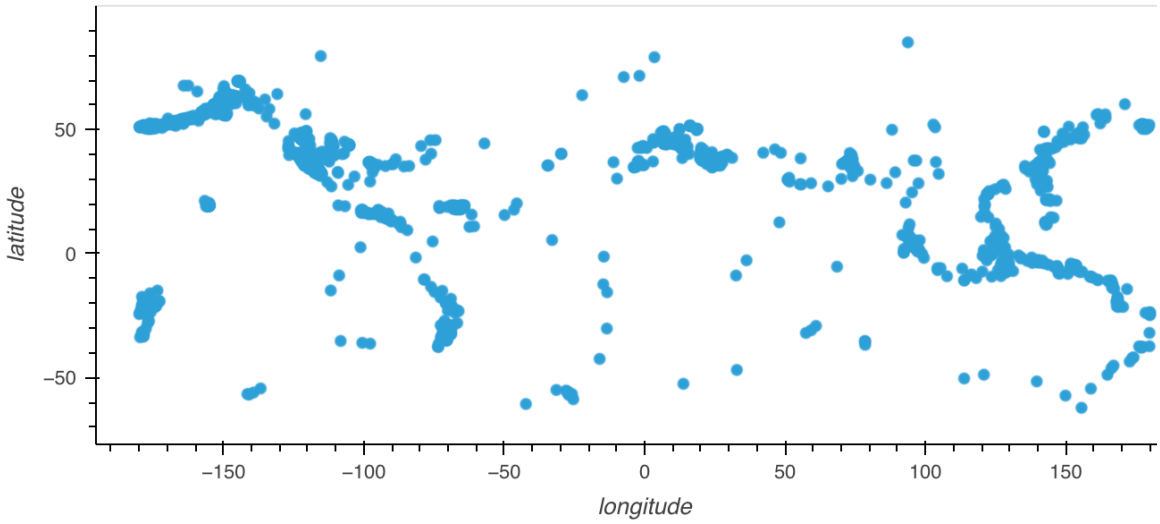
```
(1393, 9)
```

Out[154…

	depth	id	latitude	longitude	mag	place	time	
620884	54.760	us2000bhij	42.599500	143.828000	4.90	61 km SW of Kushiro, Japan	2017-11-03T03:45:12.410Z	ea
338297	10.731	ci14461224	33.974833	-116.980333	2.62	5km N of Beaumont, CA	2009-05-21T22:05:43.340Z	ea
119378	4.600	usp000ca51	44.297000	11.487000	2.10	7 km NW of Fontanelice, Italy	2003-10-11T11:18:22.300Z	ea
385612	7.400	ak010d22kdri	52.611700	-169.282500	2.20	46 km SW of Nikolski, Alaska	2010-10-11T23:41:29.104Z	ea
540705	9.800	nn00507286	41.870000	-119.616900	2.50	44 km E of Fort Bidwell, California	2015-08-20T21:38:51.403Z	ea

In [155…
```
small_df.hvplot.scatter(x="longitude", y="latitude")
```

Out[155…



`RandomForestRegressor` для себя требует числовые значения, поэтому немного предобработаем набор данны Преобразуем колонку даты в совокупность числовых признаков.

Также избавимся от неинтересующих на данном этапе колонок "id" и "place": первое является неинформативным, второе - лишь интерпретация местоположения, выраженного в признаках широты и долготы.

In [156…
```
# small_df.set_index("time", inplace=True)
small_df["dt"] = pd.to_datetime(small_df["time"])
small_df.head()
```

Out[156…

	depth	id	latitude	longitude	mag	place	time	
620884	54.760	us2000bhij	42.599500	143.828000	4.90	61 km SW of Kushiro, Japan	2017-11-03T03:45:12.410Z	ea
338297	10.731	ci14461224	33.974833	-116.980333	2.62	5km N of Beaumont, CA	2009-05-21T22:05:43.340Z	ea
119378	4.600	usp000ca51	44.297000	11.487000	2.10	7 km NW of Fontanelice, Italy	2003-10-11T11:18:22.300Z	ea
385612	7.400	ak010d22kdri	52.611700	-169.282500	2.20	46 km SW of Nikolski, Alaska	2010-10-11T23:41:29.104Z	ea
540705	9.800	nn00507286	41.870000	-119.616900	2.50	44 km E of Fort Bidwell, California	2015-08-20T21:38:51.403Z	ea

In [157…

```
# День
small_df["day"] = small_df["dt"].dt.day
# Месяц
small_df["month"] = small_df["dt"].dt.month
# Год
small_df["year"] = small_df["dt"].dt.year
# Часы
small_df["hour"] = small_df["dt"].dt.hour
# Минуты
small_df["minute"] = small_df["dt"].dt.minute
small_df.head()
```

Out[157…

	depth	id	latitude	longitude	mag	place	time	
620884	54.760	us2000bhij	42.599500	143.828000	4.90	61 km SW of Kushiro, Japan	2017-11-03T03:45:12.410Z	ea
338297	10.731	ci14461224	33.974833	-116.980333	2.62	5km N of Beaumont, CA	2009-05-21T22:05:43.340Z	ea
119378	4.600	usp000ca51	44.297000	11.487000	2.10	7 km NW of Fontanelice, Italy	2003-10-11T11:18:22.300Z	ea
385612	7.400	ak010d22kdri	52.611700	-169.282500	2.20	46 km SW of Nikolski, Alaska	2010-10-11T23:41:29.104Z	ea
540705	9.800	nn00507286	41.870000	-119.616900	2.50	44 km E of Fort Bidwell, California	2015-08-20T21:38:51.403Z	ea

Подготовим выборку для обучения, убрав неинтересующие признаки и уже конвертированные в совместимый с регрессором формат "time" и "dt".

In [158…

```
from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS  # noqa
from sklearn.model_selection import train_test_split  # noqa

X_train, X_test, y_train, y_test = train_test_split(
```

```
    small_df.drop(
        columns=["mag", "mag_reciprocal", "id", "place", "time", "dt", "type"]
    ),
    small_df["mag_reciprocal"],
    test_size=0.2,
    random_state=42,
)
print(X_train.shape, X_test.shape, y_train.shape, X_test.shape)
```

```
(1114, 8) (279, 8) (1114,) (279, 8)
```

In [159…

```
# Тестовый прогон регрессора
from sklearn.ensemble import RandomForestRegressor  # noqa

reg = RandomForestRegressor(random_state=42)
reg.fit(X_train, y_train)
reg.predict(X_test)
```

Out[159…

```
array([0.36784394, 0.22642899, 0.22702345, 0.21806412, 0.36549276,
       0.27150573, 0.36191293, 0.22770529, 0.22406893, 0.28254225,
       0.39304596, 0.21474223, 0.22986102, 0.41016969, 0.38012159,
       0.21978851, 0.41985445, 0.42456014, 0.39305778, 0.40863922,
       0.2248907 , 0.43162724, 0.41382334, 0.21896877, 0.35116086,
       0.43273281, 0.402717  , 0.41389031, 0.38382425, 0.34830891,
       0.41380505, 0.21230029, 0.22169474, 0.43265416, 0.22623527,
       0.23940745, 0.31775243, 0.35305401, 0.22247287, 0.35778916,
       0.23818595, 0.21250362, 0.32884794, 0.24861008, 0.39563025,
       0.38696181, 0.42665516, 0.3096671 , 0.41301052, 0.42494972,
       0.2313281 , 0.42245035, 0.40729826, 0.23005974, 0.43121324,
       0.21103908, 0.39412148, 0.22972019, 0.43861607, 0.37337476,
       0.39534325, 0.39830368, 0.38915229, 0.22486401, 0.38863573,
       0.41114848, 0.41666295, 0.40370405, 0.25720987, 0.22284635,
       0.37456916, 0.40305958, 0.29807037, 0.22172494, 0.41436424,
       0.41701002, 0.427902  , 0.4149095 , 0.22200363, 0.22628966,
       0.33437249, 0.2345233 , 0.34234351, 0.24239761, 0.22446631,
       0.37158701, 0.44451967, 0.42509044, 0.41107488, 0.22410457,
       0.22583592, 0.24331528, 0.22480535, 0.4152927 , 0.3815971 ,
       0.41745867, 0.3875207 , 0.3596622 , 0.22255138, 0.38191768,
       0.23577103, 0.22109346, 0.41103264, 0.35321351, 0.22950299,
       0.36900547, 0.4305989 , 0.23000277, 0.40442499, 0.22333948,
       0.4464193 , 0.35030338, 0.35398298, 0.24874447, 0.35820832,
       0.41738463, 0.39254228, 0.39257093, 0.40862833, 0.22689946,
       0.42261471, 0.37061235, 0.28039028, 0.2307118 , 0.39041837,
       0.24219359, 0.40831777, 0.44383774, 0.22940919, 0.25348145,
       0.31124282, 0.44787502, 0.41042919, 0.42773519, 0.38594565,
       0.21225139, 0.38583446, 0.40086853, 0.37530653, 0.37033619,
       0.24327301, 0.41788552, 0.21595049, 0.39795414, 0.33733498,
       0.40593439, 0.23293153, 0.41647044, 0.30713527, 0.30390696,
       0.41862371, 0.42037442, 0.37030344, 0.40027991, 0.34519001,
       0.22877653, 0.36518686, 0.21438698, 0.4186156 , 0.22322785,
       0.23097941, 0.4073025 , 0.39451333, 0.29341791, 0.35761759,
       0.23323987, 0.23834089, 0.35453103, 0.42931846, 0.38020384,
       0.44435428, 0.28114018, 0.37900023, 0.39335177, 0.3663998 ,
       0.21758761, 0.38546463, 0.32753759, 0.22434724, 0.40936301,
       0.23526383, 0.41714126, 0.22019242, 0.37407881, 0.23316595,
       0.3836059 , 0.22469272, 0.37711006, 0.37789079, 0.33392831,
       0.38391321, 0.41385244, 0.30023263, 0.37523341, 0.21916016,
       0.22579326, 0.40771357, 0.42373978, 0.40827742, 0.22729156,
       0.22663738, 0.22351498, 0.43629922, 0.35638907, 0.22391268,
       0.30037982, 0.395294  , 0.3759385 , 0.36348126, 0.39676636,
       0.22407976, 0.22650266, 0.37902509, 0.39359345, 0.22483654,
       0.21519277, 0.32230946, 0.22391402, 0.39617357, 0.3140406 ,
       0.45355016, 0.44894222, 0.22695116, 0.40958729, 0.40197874,
       0.3905706 , 0.37137156, 0.40183524, 0.34961373, 0.41114931,
       0.40024092, 0.40694731, 0.38867415, 0.22646292, 0.43669449,
       0.38295733, 0.22025102, 0.23152028, 0.41300917, 0.23345865,
       0.42056162, 0.39058922, 0.34738515, 0.39319035, 0.40913767,
       0.26875444, 0.37315325, 0.42660769, 0.37981589, 0.29753543,
       0.33745994, 0.27311495, 0.22613495, 0.30851472, 0.38223733,
       0.43133968, 0.25234847, 0.21495852, 0.3893863 , 0.3500694 ,
       0.36873235, 0.2175862 , 0.4329313 , 0.23077523, 0.41561895,
       0.41750313, 0.22631834, 0.23005191, 0.22407918, 0.32605933,
       0.34181463, 0.37230517, 0.36407865, 0.33785219, 0.37078823,
       0.36589723, 0.42504078, 0.21785763, 0.41203949])
```

Возьмём малое подмножество признаков, чтобы уменьшить время обучения модели.

In [166…

```
# Заглушим уведомления о будущих изменениях в sklearn, которые ещё не
# поддержали в библиотеке mlxtend.
import warnings  # noqa

warnings.filterwarnings("ignore", category=FutureWarning)

efs = EFS(reg, min_features=3, max_features=4, scoring="neg_mean_squared_error", cv=
```

```
efs.fit(X_train, y_train)

print("Лучшая MSE оценка: %.2f" % efs.best_score_ * (-1))
print("Лучшее подмножество признаков:", efs.best_idx_)
```

```
Features: 126/126
Лучшее подмножество признаков: (1, 2, 6, 7)
```

In [173…

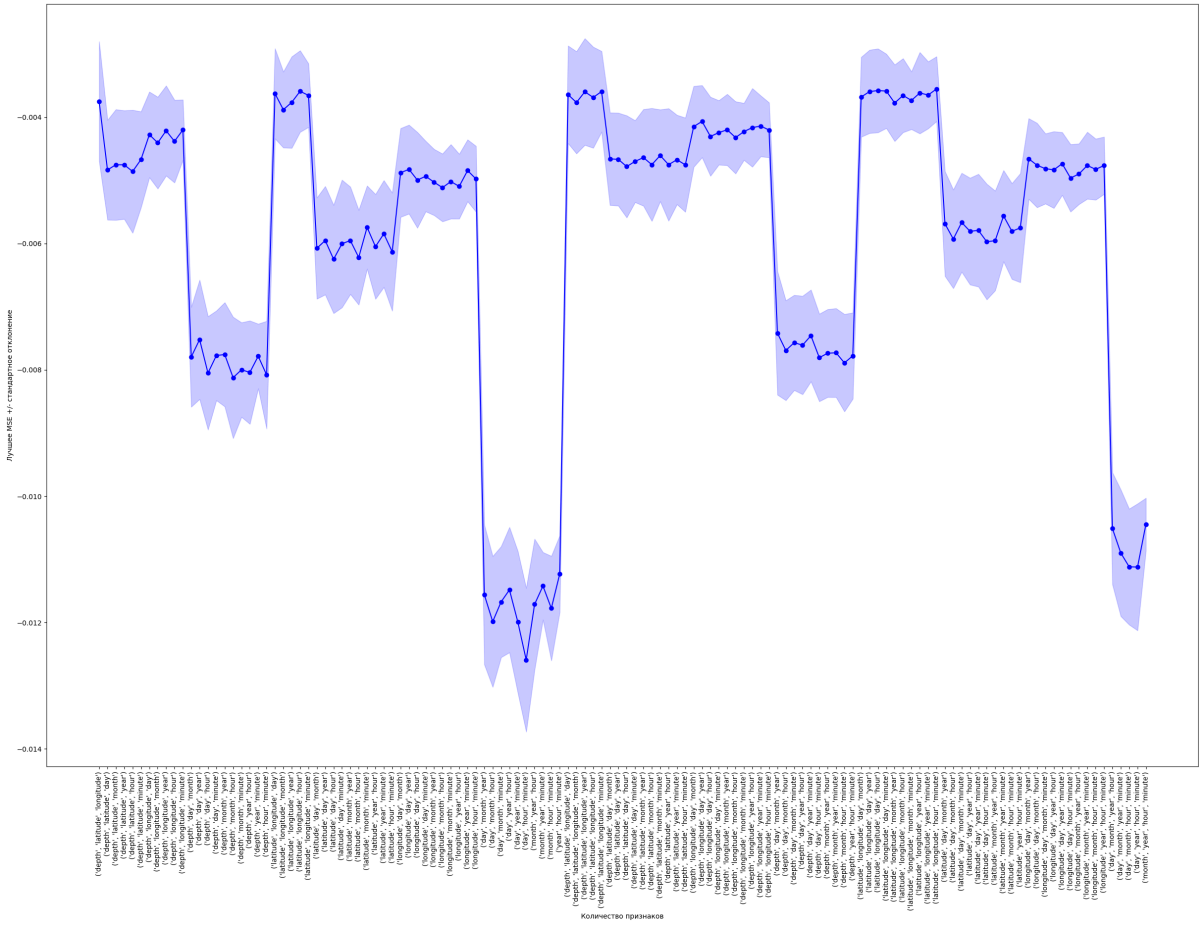
```
metric_dict = efs.get_metric_dict()

fig = plt.figure(figsize=(30, 20))
k_feat = sorted(metric_dict.keys())
avg = [metric_dict[k]["avg_score"] for k in k_feat]

upper, lower = [], []
for k in k_feat:
    upper.append(metric_dict[k]["avg_score"] + metric_dict[k]["std_dev"])
    lower.append(metric_dict[k]["avg_score"] - metric_dict[k]["std_dev"])

plt.fill_between(k_feat, upper, lower, alpha=0.2, color="blue", lw=1)

plt.plot(k_feat, avg, color="blue", marker="o")
plt.ylabel("Лучшее MSE +/- стандартное отклонение")
plt.xlabel("Количество признаков")
feature_min = len(metric_dict[k_feat[0]]["feature_idx"])
feature_max = len(metric_dict[k_feat[-1]]["feature_idx"])
plt.xticks(k_feat, [str(metric_dict[k]["feature_names"]) for k in k_feat], rotation=
plt.show()
```



Вывод

Итого, с условием взять из исходного набора данных минимум 3, а максимум 4 признака, наилучший результат показывают наборы из признаков, имеющих как минимум широту, долготу и глубину.

Это вполне соотносится с реальностью: большинство землетрясений происходят в одном и том же месте, ввиду географических особенностей области.