We will use CatBoost as base model. At first, let's try to put data to the model without transformations. After that we will try to replace missing values by -1. After that we will try to fill gaps with KNNImputer from Scikit Learn. Next idea to try is filling the missing values with average values depending on class labels.

Ввод [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import missingno as msno
from matplotlib import pyplot as plt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
```

Ввод [3]:
```
from numpy.lib.function_base import average
from sklearn.pipeline import Pipeline
from sklearn.impute import KNNImputer, SimpleImputer
# from imblearn.over_sampling import SMOTE

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from tqdm import tqdm
```

Ввод [78]:
```
pip install catboost
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) https://us-python.pkg.dev/colab-wheels/public/simple/ (ht
tps://us-python.pkg.dev/colab-wheels/public/simple/)
Requirement already satisfied: catboost in /usr/local/lib/python3.7/dist-packages (1.0.6)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from catboost) (3.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from catboost) (1.7.3)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from catboost) (1.15.0)
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.7/dist-packages (from catboost) (1.3.5)
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (from catboost) (0.10.1)
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (from catboost) (5.5.0)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.7/dist-packages (from catboost) (1.21.6)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24.0->catboost) (2022.2.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24.0->catboost)
 (2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotli
b->catboost) (3.0.9)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->catboost) (1.4.4)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib->c
atboost) (4.1.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.7/dist-packages (from plotly->catboost) (8.0.1)
```

Ввод [4]:
```
from catboost import CatBoostClassifier
```

Ввод [79]:
```
#pip install imbalanced-learn
#pip install missingno
#pip install catboost
```

Ввод [5]:
```
from matplotlib import rcParams

# figure size in inches
rcParams['figure.figsize'] = 15,12
```

Ввод [ ]:
```
from google.colab import drive
drive.mount('drive')
```

```
Mounted at drive
```

Ввод [6]:
```
data = pd.read_csv('../CSV/Norwey data.csv', sep=';')
# data
```

Ввод [7]:
```
features = data.loc[:, "CALI":"RXO"]
# features
```

Ввод [8]:
```
targets = data['FORCE_2020_LITHOFACIES_LITHOLOGY']

label_encoder = preprocessing.LabelEncoder()
encoded_targets = label_encoder.fit_transform(targets)
```

Ввод [9]:
```
gaps_pct_threshold = 50

# Percent of missing values in data
nans_pct = (features.isnull().sum()/features.shape[0]*100).astype('int').sort_values(ascending=False)
# We select features with missing values less than n %
good_columns = nans_pct[nans_pct < gaps_pct_threshold].index.to_list()
good_features = features[good_columns]
good_targets = encoded_targets[good_features.index]
```

Ввод [11]:
```
def fill_nans_with_value_from_distribution(data, target):
    data_n_target = data.join(target)
    target_col_name = target.name
    # for class_name in np.unique(target):
    for col_name in data.columns:
        data_n_target[col_name] = data_n_target[col_name].fillna(data_n_target.groupby(target_col_name)[col_name].transform('mean'
    return data_n_target.drop(columns=[target_col_name])
```

1. CatBoost and Virgin Data

Ввод [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(features, encoded_targets, test_size=0.3, random_state=1, shuffle=True, strati

catboost_model = CatBoostClassifier(iterations=70,
                            learning_rate=0.5,
                            depth=2,
                            custom_metric=['TotalF1:average=Macro']
                            )

catboost_model.fit(X_train, y_train, verbose=False)

score_1 = catboost_model.score(X_test, y_test)

print(f"Score: {score_1}")
```

```
Score: 0.7648723921698173
```

2. CatBoost + normalized data and replaced missing values

Ввод [ ]:
```
scaler = preprocessing.StandardScaler()

imputer = SimpleImputer(strategy='constant', fill_value=-1)

X_train, X_test, y_train, y_test = train_test_split(features, encoded_targets, random_state=0)

catboost_model = CatBoostClassifier(iterations=70,
                            learning_rate=0.5,
                            depth=2,
                            custom_metric=['TotalF1:average=Macro']
                            )

pipeline_2 = Pipeline(steps=[('scaler', scaler), ('imputer', imputer), ('model', catboost_model)])

pipeline_2.fit(X_train, y_train)

score_2 = pipeline_2.score(X_test, y_test)

print(f"Score: {score_2}")
```

```
0:      learn: 1.2678461        total: 1.56s    remaining: 1m 47s
1:      learn: 1.1104466        total: 2.81s    remaining: 1m 35s
2:      learn: 1.0550017        total: 4s       remaining: 1m 29s
3:      learn: 0.9970682        total: 5.34s    remaining: 1m 28s
4:      learn: 0.9616488        total: 6.73s    remaining: 1m 27s
5:      learn: 0.9410390        total: 7.99s    remaining: 1m 25s
6:      learn: 0.9259078        total: 9.26s    remaining: 1m 23s
7:      learn: 0.9116644        total: 10.6s    remaining: 1m 21s
8:      learn: 0.9120892        total: 11.7s    remaining: 1m 19s
9:      learn: 0.8929877        total: 13.8s    remaining: 1m 22s
10:     learn: 0.8782309        total: 15.4s    remaining: 1m 22s
11:     learn: 0.8694416        total: 16.8s    remaining: 1m 21s
12:     learn: 0.8624586        total: 18.2s    remaining: 1m 19s
13:     learn: 0.8527893        total: 19.7s    remaining: 1m 18s
14:     learn: 0.8716393        total: 20.8s    remaining: 1m 16s
15:     learn: 0.8491921        total: 21.9s    remaining: 1m 13s
16:     learn: 0.8407420        total: 23.6s    remaining: 1m 13s
17:     learn: 0.8344396        total: 24.8s    remaining: 1m 11s
18:     learn: 0.8307386        total: 26s      remaining: 1m 9s
19:     learn: 0.8231496        total: 27.4s    remaining: 1m 8s
```

We did not get significantly high score from both ideas. Let's move further

3. CatBoost + Features that have less than 50% of missing values

Ввод [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(good_features, good_targets, test_size=0.3, random_state=1, shuffle=True, stra

catboost_model = CatBoostClassifier(iterations=70,
                            learning_rate=0.5,
                            depth=2,
                            custom_metric=['TotalF1:average=Macro']
                            )

catboost_model.fit(X_train, y_train, verbose=False)

score_3 = catboost_model.score(X_test, y_test)

print(f"Score: {score_3}")
```

```
Score: 0.7630840030300097
```

Removing of features with huge number of missing values decreases the score

4. CatBoost and Stochastic KNNImputer

Ввод [ ]:
```
def stochastic_knn_imputer(knn_imputer, data, num_butches, fit_or_transform):
    data_after_knn_processing = None
    previous_chunk = None
    for chunk in tqdm(np.array_split(data, num_butches)):   # 24579 rows in chunk for X_train
        if previous_chunk is not None:
            chunk = np.concatenate((previous_chunk, chunk))
            previous_chunk = None
        if ((np.isnan(chunk).sum(axis=0)/chunk.shape[0]*100)==100).any():   # if any column in chunk has 100% nans then we concate
            previous_chunk = chunk
            continue
        imputted_chunk = knn_imputer.fit_transform(chunk) if fit_or_transform == 'fit' else knn_imputer.transform(chunk)
        if data_after_knn_processing is None:
            data_after_knn_processing = imputted_chunk
        else:
            data_after_knn_processing = np.concatenate((data_after_knn_processing, imputted_chunk))
    return data_after_knn_processing
```

Ввод [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(good_features, good_targets, test_size=0.3, random_state=1, shuffle=False)

imputer = KNNImputer(n_neighbors=3, weights='uniform', metric='nan_euclidean', missing_values=np.nan)

# fit stochastic knn imputer
X_train_imputted = stochastic_knn_imputer(imputer, X_train, 100, 'fit')
X_test_impitted = stochastic_knn_imputer(imputer, X_test, 14, 'transform')

catboost_model = CatBoostClassifier(iterations=70,
                            learning_rate=0.5,
                            depth=2,
                            custom_metric=['TotalF1:average=Macro']
                            )

catboost_model.fit(X_train_imputted, y_train)

score_4 = catboost_model.score(X_test_impitted, y_test)
print(f"Score: {score_4}")
```

```
  0%|          | 0/10 [00:00<?, ?it/s]
```

5. CatBoost + Filling missing values with average values depending on a class label

Ввод [12]:
```
X_train, X_test, y_train, y_test = train_test_split(good_features, good_targets, test_size=0.3, random_state=1, shuffle=True, stra

prepared_train_target = pd.Series(y_train, index=X_train.index, name=targets.name)
prepared_test_target = pd.Series(y_test, index=X_test.index, name=targets.name)

X_train_filled = fill_nans_with_value_from_distribution(X_train, prepared_train_target)
X_train_filled = X_train_filled.fillna(X_train_filled.mean())
y_train_prepared = prepared_train_target[X_train_filled.index]

X_test_filled = fill_nans_with_value_from_distribution(X_test, prepared_test_target)
X_test_filled = X_test_filled.fillna(X_test_filled.mean())
y_test_prepared = prepared_test_target[X_test_filled.index]

catboost_model = CatBoostClassifier(iterations=70,
                            learning_rate=0.5,
                            depth=2,
                            custom_metric=['TotalF1:average=Macro']
                            )

catboost_model.fit(X_train_filled, y_train_prepared, verbose=False)

score_5 = catboost_model.score(X_test_filled, y_test_prepared)

print(f"Score: {score_5}")
```

```
Score: 0.9092762719490594
```

Filling NANs with the average value depending on class label helps to improve data significantly. Score rises from .77 to .90

6. Data Oversampling

We have minority class with 103 samples. Let us oversample data to help the model to get more info from data

Ввод [ ]:
```
pd.Series(good_targets).value_counts()
```

Out[60]:
```
1     720803
0     168937
2     150455
3      56320
6      33329
11     15245
4      10513
8       8213
9       3820
5       1688
7       1085
10       103
dtype: int64
```

Ввод [ ]:
```
strategy = {10:400}  # increasing the minority class size
oversample = SMOTE(sampling_strategy=strategy)
X_train_oversampled, y_train_oversampled = oversample.fit_resample(X_train_filled, y_train_prepared)

catboost_model = CatBoostClassifier(iterations=70,
                            learning_rate=0.5,
                            depth=2,
                            custom_metric=['TotalF1:average=Macro']
                            )

catboost_model.fit(X_train_oversampled, y_train_oversampled, verbose=False)

score_6 = catboost_model.score(X_test_filled, y_test_prepared)

print(f"Score: {score_6}")
```

```
Score: 0.9044863507179186
```

Ввод [ ]:
```
pd.Series(y_train_oversampled).value_counts()
```

Out[64]:
```
1     504562
0     118256
2     105318
3      39424
6      23330
11     10671
4       7359
8       5749
9       2674
5       1182
7        760
10       300
Name: FORCE_2020_LITHOFACIES_LITHOLOGY, dtype: int64
```

Oversampling method did not help us to improve the data.

Model tuning

Ввод [13]:
```
pip install scikit-optimize
```

```
Collecting scikit-optimize
  Downloading scikit_optimize-0.9.0-py2.py3-none-any.whl (100 kB)
Requirement already satisfied: numpy>=1.13.3 in c:\programdata\anaconda3\lib\site-packages (from scikit-optimize) (1.20.1)
Collecting pyaml>=16.9
  Downloading pyaml-21.10.1-py2.py3-none-any.whl (24 kB)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-optimize) (0.24.1)
Requirement already satisfied: scipy>=0.19.1 in c:\programdata\anaconda3\lib\site-packages (from scikit-optimize) (1.6.2)
Requirement already satisfied: joblib>=0.11 in c:\programdata\anaconda3\lib\site-packages (from scikit-optimize) (1.0.1)
Requirement already satisfied: PyYAML in c:\programdata\anaconda3\lib\site-packages (from pyaml>=16.9->scikit-optimize) (5.4.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn>=0.20.0->sci
kit-optimize) (2.1.0)
Installing collected packages: pyaml, scikit-optimize
Successfully installed pyaml-21.10.1 scikit-optimize-0.9.0
Note: you may need to restart the kernel to use updated packages.
```

Ввод [37]:
```
from skopt import gp_minimize
from skopt.space import Real, Integer, Categorical
from skopt.utils import use_named_args

data = (X_train_filled, y_train_prepared, X_test_filled, y_test_prepared)

space = [Integer(100, 500, name='iterations'),
         Real(0.01, 1, 'log-uniform', name='learning_rate'),
         Integer(2, 10, name='depth'),
         Integer(1, 10, name='l2_leaf_reg'),
         # Categorical(['TotalF1:average=Macro'], name='custom_metric'),
        ]

@use_named_args(space)
def objective(**params):
    X_train, y_train, X_test, y_test = data
    model = CatBoostClassifier(**params, custom_metric=['TotalF1:average=Macro'])
    model.fit(X_train, y_train)
    return -model.score(X_test, y_test)

result = gp_minimize(objective, space, n_calls=20, random_state=1, verbose=True, n_jobs=-2)
print('Best Score: %.3f' % (-result.fun))
print('Best Parameters:' + (" {:.5f}" * len(result.x)).format(*result.x))
```

```
434:    learn: 0.0457366        total: 17m 27s  remaining: 2m 36s
435:    learn: 0.0456715        total: 17m 30s  remaining: 2m 34s
436:    learn: 0.0455773        total: 17m 32s  remaining: 2m 31s
437:    learn: 0.0454926        total: 17m 35s  remaining: 2m 29s
438:    learn: 0.0454310        total: 17m 37s  remaining: 2m 26s
439:    learn: 0.0453593        total: 17m 40s  remaining: 2m 24s
440:    learn: 0.0453117        total: 17m 42s  remaining: 2m 22s
441:    learn: 0.0452013        total: 17m 45s  remaining: 2m 19s
442:    learn: 0.0451379        total: 17m 47s  remaining: 2m 17s
443:    learn: 0.0450686        total: 17m 50s  remaining: 2m 14s
444:    learn: 0.0450261        total: 17m 52s  remaining: 2m 12s
445:    learn: 0.0449646        total: 17m 54s  remaining: 2m 10s
446:    learn: 0.0448774        total: 17m 57s  remaining: 2m 7s
447:    learn: 0.0448348        total: 17m 59s  remaining: 2m 5s
448:    learn: 0.0447943        total: 18m 2s   remaining: 2m 2s
449:    learn: 0.0447266        total: 18m 4s   remaining: 2m
450:    learn: 0.0446702        total: 18m 7s   remaining: 1m 58s
451:    learn: 0.0445970        total: 18m 9s   remaining: 1m 55s
452:    learn: 0.0445162        total: 18m 12s  remaining: 1m 53s
```

Ввод [39]:
```
catboost_model = CatBoostClassifier(iterations=500,
                            learning_rate=0.16,
                            depth=10,
                            l2_leaf_reg=6,
                            custom_metric=['TotalF1:average=Macro']
                            )

catboost_model.fit(X_train_filled, y_train_prepared, verbose=False)

score_7= catboost_model.score(X_test_filled, y_test_prepared)

print(f"Score: {score_7}")
```

```
Score: 0.9701270667570354
```

As conclusion, we can say that filling the missing values with average values depending on class label gave good result rising score on test from 77% to 91%.
Model tuning rised the accuracy of prediction to 97%.
Unfortunately, oversampling did not increase the score on test. Methods which being used also are 1) CatBoost and Virgin Data, 2) CatBoost + normalized data and replaced missing values(-1), 3) removing features that contains more than 50% of missing values and 4) Stochastic KNN Imputer of gaps.

Ввод [ ]: