There is described path of selecting and training model

```
import os
import pandas as pd
import numpy as np
import torch

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from transformers import AutoTokenizer, AutoModel
import catboost
import nltk
from nltk.tokenize import word_tokenize
import string
import gensim.downloader
from nltk.corpus import stopwords
import re
```

Data load and prepare code (initial)

```
def load_text(path):
    with open(path, 'r') as f:
        return f.read()
```

```
def load_data(path):
    train_path = os.path.join(path, 'train')
    test_path = os.path.join(path, 'test')

    train_pos_path = os.path.join(train_path, 'pos')
    train_neg_path = os.path.join(train_path, 'neg')

    test_pos_path = os.path.join(test_path, 'pos')
    test_neg_path = os.path.join(test_path, 'neg')

    train_pos = pd.DataFrame([[int(os.path.splitext(x)[0].split('_')[1]), os.path.join(tr
    train_pos["Full text"] = train_pos.apply(lambda x: load_text(x["Full path"]), axis=1)

    train_neg = pd.DataFrame([[int(os.path.splitext(x)[0].split('_')[1]), os.path.join(tr
    train_neg["Full text"] = train_neg.apply(lambda x: load_text(x["Full path"]), axis=1)

    test_pos = pd.DataFrame([[int(os.path.splitext(x)[0].split('_')[1]), os.path.join(tes
    test_pos["Full text"] = test_pos.apply(lambda x: load_text(x["Full path"]), axis=1)

    test_neg = pd.DataFrame([[int(os.path.splitext(x)[0].split('_')[1]), os.path.join(tes
    test_neg["Full text"] = test_neg.apply(lambda x: load_text(x["Full path"]), axis=1)

    train = pd.concat([train_pos, train_neg])#.sample(frac=1)
    test = pd.concat([test_pos, test_neg])#.sample(frac=1)

    train = train.drop(columns=["Full path"])

    test = test.drop(columns=["Full path"])

    return train, test



train_data, test_data = load_data('aclImdb')
```

First tested model - BERT-base + CatBoostRegressor

```
tokenizer = AutoTokenizer.from_pretrained("google-bert/bert-base-uncased")
model = AutoModel.from_pretrained("google-bert/bert-base-uncased")
model.cuda()
device = torch.device("cuda:0")
```

```
/home/ded/miniconda3/envs/ml/lib/python3.12/site-packages/transformers/tokenization_u
  warnings.warn(
```

```
X = train_data['Full text'].reset_index(drop=True)
y = train_data['Mark'].reset_index(drop=True)
```

```
def tokenize(x):
    x = re.sub(r'[^\w\s]', '', x)
    stopwords_set = set(stopwords.words('english'))
    wnl = nltk.WordNetLemmatizer()
    x = ' '.join([wnl.lemmatize(word) for word in word_tokenize(x.lower()) if (word not i
    t = tokenizer(x, padding=True, truncation=True, return_tensors='pt')
    return t


tokenized_train = pd.concat([pd.DataFrame(X.apply(lambda x: tokenize(x))), y], axis=1)


tokenized_train
```

	Full text	Mark
0	[input_ids, token_type_ids, attention_mask]	7
1	[input_ids, token_type_ids, attention_mask]	7
2	[input_ids, token_type_ids, attention_mask]	9
3	[input_ids, token_type_ids, attention_mask]	7
4	[input_ids, token_type_ids, attention_mask]	7
...	...	...
24995	[input_ids, token_type_ids, attention_mask]	1
24996	[input_ids, token_type_ids, attention_mask]	3
24997	[input_ids, token_type_ids, attention_mask]	3
24998	[input_ids, token_type_ids, attention_mask]	2
24999	[input_ids, token_type_ids, attention_mask]	3

25000 rows × 2 columns

```
embeddings = []
labels = []
for index, row in tokenized_train.iterrows():
    row_data = row["Full text"]

    labels.append(row["Mark"])
    with torch.no_grad():
        model_output = model(**{k: v.to(model.device) for k, v in row_data.items()})
        embedding = model_output.last_hidden_state[:, 0, :]
        embedding = torch.nn.functional.normalize(embedding)
    embeddings.append(embedding[0].cpu().numpy())


features = pd.DataFrame(embeddings)
labels = np.array(labels)


features
```

	0	1	2	3	4	5	6	7
0	-0.023313	0.005784	0.034300	-0.003292	-0.015695	-0.015810	0.013644	-0.008273
1	-0.000374	0.008638	0.026132	0.002072	-0.019049	-0.029332	0.011774	0.004022
2	-0.014271	0.008412	0.022101	-0.013482	0.000115	-0.037497	0.013218	0.010021
3	-0.019618	0.000270	0.000530	-0.008883	-0.017093	-0.014391	0.016084	-0.015722
4	-0.014457	0.004122	0.022858	-0.018852	-0.013125	0.001092	0.010123	0.012368
...	...	...	...	...	...	...	...	...
24995	-0.015981	0.006766	0.012822	0.003048	-0.024360	-0.002777	0.007232	0.002981
24996	-0.002732	-0.002021	0.024036	0.002596	-0.026290	-0.028555	0.009510	0.010641
24997	0.002663	-0.022319	0.020631	-0.012436	-0.026644	-0.023603	0.027566	0.011035
24998	0.014351	0.010978	0.012884	-0.006181	-0.023091	-0.027680	0.008688	0.027690
24999	-0.006602	0.017689	0.007408	-0.001567	-0.022781	-0.020376	0.025251	0.025730

25000 rows × 768 columns

```
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, shuf

X_test, X_eval, y_test, y_eval = train_test_split(X_test, y_test, test_size=0.4, shuffle=

pool_train = catboost.Pool(data=X_train, label=y_train)
pool_eval = catboost.Pool(data=X_eval, label=y_eval)

ctb = catboost.CatBoostRegressor(verbose=100, task_type="GPU", devices="0")

ctb.fit(pool_train, eval_set=pool_eval, plot=True)
```

```
MetricVisualizer(layout=Layout(align_self='stretch', height='500px'))
Learning rate set to 0.085827
0:      learn: 3.4024723        test: 3.4387170 best: 3.4387170 (0)     total: 183ms
100:    learn: 2.5369648        test: 2.7163700 best: 2.7163700 (100)   total: 9.35s
200:    learn: 2.4014354        test: 2.6825165 best: 2.6825165 (200)   total: 17.3s
300:    learn: 2.2987353        test: 2.6643957 best: 2.6643957 (300)   total: 25.1s
400:    learn: 2.2113170        test: 2.6536597 best: 2.6535752 (399)   total: 32.9s
500:    learn: 2.1345129        test: 2.6473017 best: 2.6470715 (497)   total: 40.7s
600:    learn: 2.0649850        test: 2.6464838 best: 2.6461021 (597)   total: 49.4s
700:    learn: 2.0047091        test: 2.6393449 best: 2.6391107 (699)   total: 58s
800:    learn: 1.9474480        test: 2.6359915 best: 2.6356588 (793)   total: 1m 6s
900:    learn: 1.8964684        test: 2.6329381 best: 2.6329381 (900)   total: 1m 14s
999:    learn: 1.8466990        test: 2.6295149 best: 2.6287587 (996)   total: 1m 22s
bestTest = 2.628758749
bestIteration = 996
Shrink model to first 997 iterations.
<catboost.core.CatBoostRegressor at 0x7ff0f863e6f0>
```

```
ctb.score(X_test, y_test)
```

```
0.45280708695702954
```

First model did not show good results. Next step - try GridCV

```
train_data, test_data = load_data('aclImdb')


X = train_data['Full text'].reset_index(drop=True)
y = train_data['Mark'].reset_index(drop=True)


tokenized_train = pd.concat([pd.DataFrame(X.apply(lambda x: tokenize(x))), y], axis=1)


tokenized_train
```

	Full text	Mark
0	[input_ids, token_type_ids, attention_mask]	7
1	[input_ids, token_type_ids, attention_mask]	7
2	[input_ids, token_type_ids, attention_mask]	9
3	[input_ids, token_type_ids, attention_mask]	7
4	[input_ids, token_type_ids, attention_mask]	7
...	...	...
24995	[input_ids, token_type_ids, attention_mask]	1
24996	[input_ids, token_type_ids, attention_mask]	3
24997	[input_ids, token_type_ids, attention_mask]	3
24998	[input_ids, token_type_ids, attention_mask]	2
24999	[input_ids, token_type_ids, attention_mask]	3

25000 rows × 2 columns

```
embeddings = []
labels = []
for index, row in tokenized_train.iterrows():
    row_data = row["Full text"]

    labels.append(row["Mark"])
    with torch.no_grad():
        model_output = model(**{k: v.to(model.device) for k, v in row_data.items()})
        embedding = model_output.last_hidden_state[:, 0, :]
        embedding = torch.nn.functional.normalize(embedding)
    embeddings.append(embedding[0].cpu().numpy())
```

```
features = pd.DataFrame(embeddings)
labels = np.array(labels)
```

```
features
```

	0	1	2	3	4	5	6	7
0	-0.023313	0.005784	0.034300	-0.003292	-0.015695	-0.015810	0.013644	-0.008273
1	-0.000374	0.008638	0.026132	0.002072	-0.019049	-0.029332	0.011774	0.004022
2	-0.014271	0.008412	0.022101	-0.013482	0.000115	-0.037497	0.013218	0.010021
3	-0.019618	0.000270	0.000530	-0.008883	-0.017093	-0.014391	0.016084	-0.015722
4	-0.014457	0.004122	0.022858	-0.018852	-0.013125	0.001092	0.010123	0.012368
...	...	...	...	...	...	...	...	...
24995	-0.015981	0.006766	0.012822	0.003048	-0.024360	-0.002777	0.007232	0.002981
24996	-0.002732	-0.002021	0.024036	0.002596	-0.026290	-0.028555	0.009510	0.010641
24997	0.002663	-0.022319	0.020631	-0.012436	-0.026644	-0.023603	0.027566	0.011035
24998	0.014351	0.010978	0.012884	-0.006181	-0.023091	-0.027680	0.008688	0.027690
24999	-0.006602	0.017689	0.007408	-0.001567	-0.022781	-0.020376	0.025251	0.025730

25000 rows × 768 columns

```
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, shuf
```

```
pool_train = catboost.Pool(data=X_train, label=y_train)
pool_test = catboost.Pool(data=X_test, label=y_test)
```

```
grid_model = catboost.CatBoostRegressor(verbose=100, task_type="GPU", devices="0")

param_dist = {
    'iterations': [10**i for i in range(3, 4)],
    'learning_rate': np.linspace(0.01, 0.2, 5),
    'depth': [4, 7],
}

grid_model.grid_search(param_grid=param_dist, X=pool_train, refit=True, verbose=100, plot
```

```
MetricVisualizer(layout=Layout(align_self='stretch', height='500px'))
0:      learn: 6.4303835        test: 6.4378137 best: 6.4378137 (0)     total: 35.2ms
100:    learn: 3.6992963        test: 3.7127893 best: 3.7127893 (100)   total: 3.48s
200:    learn: 3.0296742        test: 3.0502958 best: 3.0502958 (200)   total: 6.87s
300:    learn: 2.8471839        test: 2.8752935 best: 2.8752935 (300)   total: 10.3s
400:    learn: 2.7697081        test: 2.8048348 best: 2.8048348 (400)   total: 13.6s
500:    learn: 2.7219422        test: 2.7627307 best: 2.7627307 (500)   total: 16.9s
600:    learn: 2.6872995        test: 2.7345162 best: 2.7345162 (600)   total: 20.4s
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[30], line 9
      1 grid_model = catboost.CatBoostRegressor(verbose=100, task_type="GPU",
devices="0")
      3 param_dist = {
      4     'iterations': [10**i for i in range(3, 4)],
      5     'learning_rate': np.linspace(0.01, 0.2, 5),
      6     'depth': [4, 7],
      7 }
----> 9 grid_model.grid_search(param_grid=param_dist, X=pool_train, refit=True,
verbose=100, plot=True)

File ~/miniconda3/envs/ml/lib/python3.12/site-packages/catboost/core.py:4211, in
CatBoost.grid_search(self, param_grid, X, y, cv, partition_random_seed,
calc_cv_statistics, search_by_train_test_split, refit, shuffle, stratified,
train_size, verbose, plot, plot_file, log_cout, log_cerr)
   4208         if not isinstance(grid[key], Iterable):
   4209             raise TypeError('Parameter grid value is not iterable (key={!r},
value={!r})'.format(key, grid[key]))
-> 4211 return self._tune_hyperparams(
   4212     param_grid=param_grid, X=X, y=y, cv=cv, n_iter=-1,
   4213     partition_random_seed=partition_random_seed,
calc_cv_statistics=calc_cv_statistics,
   4214     search_by_train_test_split=search_by_train_test_split, refit=refit,
shuffle=shuffle,
   4215     stratified=stratified, train_size=train_size, verbose=verbose,
plot=plot, plot_file=plot_file,
   4216     log_cout=log_cout, log_cerr=log_cerr,
   4217 )

File ~/miniconda3/envs/ml/lib/python3.12/site-packages/catboost/core.py:4100, in
CatBoost._tune_hyperparams(self, param_grid, X, y, cv, n_iter,
partition_random_seed, calc_cv_statistics, search_by_train_test_split, refit,
shuffle, stratified, train_size, verbose, plot, plot_file, log_cout, log_cerr)
   4097     stratified = isinstance(loss_function, STRING_TYPES) and
is_cv_stratified_objective(loss_function)
   4099 with plot_wrapper(plot, plot_file, 'Hyperparameters search plot',
[ get_train_dir(params)]):
```

```
grid_model.score(X_test, y_test)
```

GridCV give no results, use default model params

I also tried some other tokenizers, they are about the same in quality

Word2Vec pretrained

```
vectors = gensim.downloader.load('glove-twitter-200')


from nltk.corpus import stopwords

def tokenize_word2vec(text):
    stopwords_set = set(stopwords.words('english'))
    wnl = nltk.WordNetLemmatizer()
    x = [wnl.lemmatize(word) for word in word_tokenize(text.lower()) if (word not in stop
    embeddings =[]
    try:
        embeddings = [vectors[word] for word in x if word in vectors]
    except KeyError as e:
        pass

    if embeddings:
        # Усредняем эмбеддинги по каждой координате
        return np.mean(embeddings, axis=0)
    return embeddings


features = []
for fea in X:
    features.append(tokenize_word2vec(fea))
features = np.array(features)


features
```

```
array([[-0.02600561,  0.0981196 , -0.06407259, ...,  0.12736613,
        -0.00164985, -0.17884152],
       [-0.01128365,  0.04778509, -0.00855022, ...,  0.15964073,
        -0.07756658, -0.11727268],
       [-0.09386162, -0.004718  , -0.13068983, ...,  0.07304465,
         0.01082644, -0.05915766],
       ...,
       [ 0.03862862,  0.14663793,  0.04064982, ...,  0.15826869,
        -0.0895642 , -0.15713716],
       [ 0.08103628,  0.24486575, -0.07451502, ...,  0.193327  ,
         0.01319468, -0.12568967],
       [-0.00223744,  0.05071123, -0.05749378, ...,  0.19464792,
        -0.04045384, -0.0445789 ]], dtype=float32)
```

```
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, shuf


X_test, X_eval, y_test, y_eval = train_test_split(X_test, y_test, test_size=0.4, shuffle=


pool_train = catboost.Pool(data=X_train, label=y_train)
pool_eval = catboost.Pool(data=X_eval, label=y_eval)


ctb_w2v_p = catboost.CatBoostRegressor(verbose=100, task_type="GPU", devices="0")
```

```
ctb_w2v_p.fit(pool_train, eval_set=pool_eval, plot=True)
```

```
MetricVisualizer(layout=Layout(align_self='stretch', height='500px'))
Learning rate set to 0.085827
0:      learn: 3.3996381        test: 3.4011909 best: 3.4011909 (0)     total: 39.6ms
100:    learn: 2.4421111        test: 2.5322818 best: 2.5322818 (100)   total: 5.06s
200:    learn: 2.3169151        test: 2.4741154 best: 2.4741154 (200)   total: 10.3s
300:    learn: 2.2276512        test: 2.4568829 best: 2.4568829 (300)   total: 14.9s
400:    learn: 2.1560159        test: 2.4445090 best: 2.4442639 (385)   total: 19.3s
500:    learn: 2.0918555        test: 2.4334249 best: 2.4334249 (500)   total: 23.9s
600:    learn: 2.0361725        test: 2.4259651 best: 2.4259651 (600)   total: 28.4s
700:    learn: 1.9886918        test: 2.4197088 best: 2.4191582 (675)   total: 33s
800:    learn: 1.9436407        test: 2.4172771 best: 2.4172771 (800)   total: 37.5s
900:    learn: 1.9002121        test: 2.4130885 best: 2.4121632 (889)   total: 42.2s
999:    learn: 1.8644514        test: 2.4126275 best: 2.4113878 (956)   total: 46.6s
bestTest = 2.411387778
bestIteration = 956
Shrink model to first 957 iterations.
<catboost.core.CatBoostRegressor at 0x7ff031f787d0>
```

```
ctb_w2v_p.score(X_test, y_test)
```

```
0.5007320811453071
```

Self-trained Word2Vec

```
from nltk.corpus import stopwords

def tokenize_word2vec2train(text):
    stopwords_set = set(stopwords.words('english'))
    wnl = nltk.WordNetLemmatizer()
    text = re.sub(r'[^\w\s]', '', text)
    x = [wnl.lemmatize(word) for word in word_tokenize(text.lower()) if (word not in stop
    return x


features = X.apply(lambda x: tokenize_word2vec2train(x))


model = gensim.models.Word2Vec(sentences=features, vector_size=200, window=5, min_count=3
```

```
def tokenize_word2vec_self(text):
    stopwords_set = set(stopwords.words('english'))
    wnl = nltk.WordNetLemmatizer()
    text = re.sub(r'[^\w\s]', '', text)
    x = [wnl.lemmatize(word) for word in word_tokenize(text.lower()) if (word not in stop
    embeddings =[]
    try:
        embeddings = [model.wv[word] for word in x if word in model.wv]
    except KeyError as e:
        pass

    if embeddings:
        # Усредняем эмбеддинги по каждой координате
        return np.mean(embeddings, axis=0)
    return embeddings



features = []
for fea in X:
    features.append(tokenize_word2vec_self(fea))


features = pd.DataFrame(features)
labels = np.array(y)


X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, shuf


X_test, X_eval, y_test, y_eval = train_test_split(X_test, y_test, test_size=0.4, shuffle=


pool_train = catboost.Pool(data=X_train, label=y_train)
pool_eval = catboost.Pool(data=X_eval, label=y_eval)


ctb_w2v_s = catboost.CatBoostRegressor(verbose=100, task_type="GPU", devices="0")


ctb_w2v_s.fit(pool_train, eval_set=pool_eval, plot=True)
```

```
MetricVisualizer(layout=Layout(align_self='stretch', height='500px'))
Learning rate set to 0.085827
0:      learn: 3.3905256        test: 3.3975001 best: 3.3975001 (0)     total: 40.4ms
100:    learn: 2.5049517        test: 2.6498708 best: 2.6496348 (99)    total: 4.71s
200:    learn: 2.3940438        test: 2.6116332 best: 2.6116332 (200)   total: 9.01s
300:    learn: 2.3143641        test: 2.5935216 best: 2.5932178 (297)   total: 13.4s
400:    learn: 2.2490555        test: 2.5803553 best: 2.5802408 (399)   total: 17.3s
500:    learn: 2.1895755        test: 2.5712406 best: 2.5711503 (499)   total: 21.3s
600:    learn: 2.1389664        test: 2.5689366 best: 2.5673231 (578)   total: 25.3s
700:    learn: 2.0978136        test: 2.5658651 best: 2.5656923 (699)   total: 29.2s
800:    learn: 2.0617054        test: 2.5626614 best: 2.5626239 (799)   total: 33.2s
900:    learn: 2.0279008        test: 2.5599774 best: 2.5597374 (885)   total: 37.1s
999:    learn: 1.9933167        test: 2.5574087 best: 2.5574087 (999)   total: 41.1s
bestTest = 2.557408719
bestIteration = 999
<catboost.core.CatBoostRegressor at 0x7ff0fa08e6f0>
```

```
ctb_w2v_s.score(X_test, y_test)
```

```
0.45533190914043264
```

To improve the quality, we will try an ensemble of models: two classifiers will complement the embedding vector with their predictions

```
#New data load func (Boolean features)
def load_data(path):
    train_path = os.path.join(path, 'train')
    test_path = os.path.join(path, 'test')

    train_pos_path = os.path.join(train_path, 'pos')
    train_neg_path = os.path.join(train_path, 'neg')

    test_pos_path = os.path.join(test_path, 'pos')
    test_neg_path = os.path.join(test_path, 'neg')

    train_pos = pd.DataFrame([[int(os.path.splitext(x)[0].split('_')[1]), os.path.join(tr
    train_pos["Full text"] = train_pos.apply(lambda x: load_text(x["Full path"]), axis=1)

    train_neg = pd.DataFrame([[int(os.path.splitext(x)[0].split('_')[1]), os.path.join(tr
    train_neg["Full text"] = train_neg.apply(lambda x: load_text(x["Full path"]), axis=1)

    test_pos = pd.DataFrame([[int(os.path.splitext(x)[0].split('_')[1]), os.path.join(tes
    test_pos["Full text"] = test_pos.apply(lambda x: load_text(x["Full path"]), axis=1)

    test_neg = pd.DataFrame([[int(os.path.splitext(x)[0].split('_')[1]), os.path.join(tes
    test_neg["Full text"] = test_neg.apply(lambda x: load_text(x["Full path"]), axis=1)

    train = pd.concat([train_pos, train_neg])#.sample(frac=1)
    test = pd.concat([test_pos, test_neg])#.sample(frac=1)

    train = train.drop(columns=["Full path"])
    train['Positive'] = train['Mark'] >= 7
    train['Negative'] = train["Mark"] <= 4

    test = test.drop(columns=["Full path"])
    test['Positive'] = test['Mark'] >= 7
    test['Negative'] = test["Mark"] <= 4



    return train, test



train_data, test_data = load_data('aclImdb')
```

```
X = train_data['Full text'].reset_index(drop=True)
y = train_data['Positive'].reset_index(drop=True)
y_2 = train_data['Negative'].reset_index(drop=True)


tokenizer = AutoTokenizer.from_pretrained("google-bert/bert-base-uncased")
model = AutoModel.from_pretrained("google-bert/bert-base-uncased")
model.cuda()
device = torch.device("cuda:0")
```

```
/home/ded/miniconda3/envs/ml/lib/python3.12/site-packages/transformers/tokenization_u
  warnings.warn(
```

```
def tokenize(x):
    x = re.sub(r'[^\w\s]', '', x)
    stopwords_set = set(stopwords.words('english'))
    wnl = nltk.WordNetLemmatizer()
    x = ' '.join([wnl.lemmatize(word) for word in word_tokenize(x.lower()) if (word not i
    t = tokenizer(x, padding=True, truncation=True, return_tensors='pt')
    return t


tokenized_embeddings = pd.DataFrame(X.apply(lambda x: tokenize(x)))


tokenized_embeddings
```

	Full text
0	[input_ids, token_type_ids, attention_mask]
1	[input_ids, token_type_ids, attention_mask]
2	[input_ids, token_type_ids, attention_mask]
3	[input_ids, token_type_ids, attention_mask]
4	[input_ids, token_type_ids, attention_mask]
...	...
24995	[input_ids, token_type_ids, attention_mask]
24996	[input_ids, token_type_ids, attention_mask]
24997	[input_ids, token_type_ids, attention_mask]
24998	[input_ids, token_type_ids, attention_mask]
24999	[input_ids, token_type_ids, attention_mask]

25000 rows × 1 columns

```
embeddings = []

for index, row in tokenized_embeddings.iterrows():
    row_data = row["Full text"]

    with torch.no_grad():
        model_output = model(**{k: v.to(model.device) for k, v in row_data.items()})
        embedding = model_output.last_hidden_state[:, 0, :]
        embedding = torch.nn.functional.normalize(embedding)
    embeddings.append(embedding[0].cpu().numpy())


features = pd.DataFrame(embeddings)
labels_pos = np.array(y)
labels_neg = np.array(y_2)


X_train, X_test, y_train_pos, y_test_pos = train_test_split(features, labels_pos, test_si


X_test, X_eval, y_test_pos, y_eval_pos = train_test_split(X_test, y_test_pos, test_size=0


pool_train_pos = catboost.Pool(data=X_train, label=y_train_pos)
pool_eval_pos = catboost.Pool(data=X_test, label=y_test_pos)


ctb_pos = catboost.CatBoostClassifier(verbose=100, task_type="GPU", devices="0")


ctb_pos.fit(pool_train_pos, eval_set=pool_eval_pos, plot=True)
```

```
MetricVisualizer(layout=Layout(align_self='stretch', height='500px'))
Learning rate set to 0.0552
0:      learn: 0.6807842        test: 0.6808469 best: 0.6808469 (0)     total: 201ms
100:    learn: 0.4528600        test: 0.4695452 best: 0.4695452 (100)   total: 9.87s
200:    learn: 0.4100656        test: 0.4495698 best: 0.4495698 (200)   total: 18.7s
300:    learn: 0.3803601        test: 0.4387194 best: 0.4386981 (298)   total: 27.6s
400:    learn: 0.3569461        test: 0.4331594 best: 0.4331594 (400)   total: 36.3s
500:    learn: 0.3370018        test: 0.4303522 best: 0.4303522 (500)   total: 45.4s
600:    learn: 0.3188604        test: 0.4272364 best: 0.4272024 (599)   total: 54s
700:    learn: 0.3044974        test: 0.4254137 best: 0.4252311 (685)   total: 1m 2s
800:    learn: 0.2910578        test: 0.4240233 best: 0.4240233 (800)   total: 1m 12s
900:    learn: 0.2779351        test: 0.4223774 best: 0.4223774 (900)   total: 1m 21s
999:    learn: 0.2664589        test: 0.4210852 best: 0.4210513 (998)   total: 1m 30s
bestTest = 0.4210513102
bestIteration = 998
Shrink model to first 999 iterations.
<catboost.core.CatBoostClassifier at 0x7f7d6d900920>
```

```
ctb_pos.score(X_eval, y_eval_pos)
```

```
0.7985
```

```
ctb_pos.save_model("model_pos")
```

```
X_train, X_test, y_train_neg, y_test_neg = train_test_split(features, labels_neg, test_si

X_test, X_eval, y_test_neg, y_eval_neg = train_test_split(X_test, y_test_neg, test_size=0


pool_train_neg = catboost.Pool(data=X_train, label=y_train_neg)
pool_eval_neg = catboost.Pool(data=X_test, label=y_test_neg)


ctb_neg = catboost.CatBoostClassifier(verbose=100, task_type="GPU", devices="0")


ctb_neg.fit(pool_train_neg, eval_set=pool_eval_neg, plot=True)
```

```
MetricVisualizer(layout=Layout(align_self='stretch', height='500px'))
Learning rate set to 0.0552
0:      learn: 0.6811795        test: 0.6817184 best: 0.6817184 (0)     total: 108ms
100:    learn: 0.4519076        test: 0.4823251 best: 0.4823251 (100)   total: 9.62s
200:    learn: 0.4087329        test: 0.4603979 best: 0.4603979 (200)   total: 18.5s
300:    learn: 0.3786442        test: 0.4501798 best: 0.4501798 (300)   total: 26.9s
400:    learn: 0.3546421        test: 0.4438354 best: 0.4438354 (400)   total: 35.6s
500:    learn: 0.3343662        test: 0.4403437 best: 0.4403437 (500)   total: 44s
600:    learn: 0.3168132        test: 0.4374014 best: 0.4373770 (598)   total: 52.4s
700:    learn: 0.3019467        test: 0.4344915 best: 0.4344915 (700)   total: 1m
800:    learn: 0.2870584        test: 0.4321711 best: 0.4321711 (800)   total: 1m 9s
900:    learn: 0.2734155        test: 0.4301605 best: 0.4301494 (898)   total: 1m 17s
999:    learn: 0.2613694        test: 0.4284285 best: 0.4284285 (999)   total: 1m 26s
bestTest = 0.4284285075
bestIteration = 999
<catboost.core.CatBoostClassifier at 0x7f7d174ee030>
```

```
ctb_neg.score(X_eval, y_eval_neg)
```

```
0.8105
```

```
ctb_neg.save_model("model_neg")


preds_pos = ctb_pos.predict(features)


preds_neg = ctb_neg.predict(features)
```