LAPIML: Launch API for ML models

This document describes working of API (application programming interface) for fitting machine learning models.

Figure 1: *Created by genereative transformer*

List of models which now are available:

- Regression task (quality of models measured by MSE)

  Ridge

- Binary classification (ROC AUC)

  LogisticRegression

  CatBoostClassifier

  DecisionTreeClassifier

- Multiclass classification (F1 score)

  RandomForestClassifier

  CatBoostClassifier

  DecisionTreeClassifier

Also it's possible to expand this list - send your pull request.

For launch API just make a few steps. In terminal:

```
cd /path/to/your/dir/with/files/flaskapi.py/and/modelapi.py/from/github
flask --app flaskapi --debug run
```

Then using Python:

```
#Let's simultae data for model classification...
X, y = make_classification(n_samples = 10**4, n_features = 4, n_informative = 2, n_classes = 2, random_state = m_seed)
data_clf = pd.DataFrame(X)
data_clf['target'] = y
X_train, X_test = train_test_split(data_clf, test_size = m_size, random_state = m_seed, shuffle = True, stratify = y)

#Check which models are available
print(requests.get('http://127.0.0.1:5000/api/get_possible_model', json = X_train.to_dict()).text)

#Creating the model
requests.post('http://127.0.0.1:5000/api/create_model', json = {'model_name': 'DecisionTreeClassifier',}).text
requests.get('http://127.0.0.1:5000/api/get_model/1').json()

#Fit the model
params = {'data': X_train.to_dict()}
requests.put('http://127.0.0.1:5000/api/fit/1', json = params).text
requests.get('http://127.0.0.1:5000/api/get_model/1').json()

#Make predictions
preds = requests.put('http://127.0.0.1:5000/api/predict_proba/1',
json = {'X': X_test.drop(columns = 'target').to_dict()}).json()

#Check the quality of the model
params = {'data': X_test.to_dict()}
requests.put('http://127.0.0.1:5000/api/get_scores/1', json = params).json()

##########################
#...and regression...
X_reg, y_reg = make_regression(n_samples = 10**4, n_features = 10, n_informative = 5, random_state = m_seed)
data_reg = pd.DataFrame(X_reg)
data_reg['target'] = y_reg
Xr_train, Xr_test = train_test_split(data_reg, test_size = m_size, random_state = m_seed, shuffle = True)

#Check which models are available
print(requests.get('http://127.0.0.1:5000/api/get_possible_model', json = Xr_train.to_dict()).text)

#Creating the model
```

```
requests.post('http://127.0.0.1:5000/api/create_model', json = {'model_name'
    : 'Ridge',}).text
requests.get('http://127.0.0.1:5000/api/get_model/2').json()

#Fit the model
params = {'data': Xr_train.to_dict(), 'alpha': 1.0}
requests.put('http://127.0.0.1:5000/api/fit/2', json = params).text
requests.get('http://127.0.0.1:5000/api/get_model/2').json()

#Make predictions
preds = requests.put('http://127.0.0.1:5000/api/predict/2',
json = {'X': Xr_test.drop(columns = 'target').to_dict()}).json()

#Check the quality of the model
params = {'data': Xr_test.to_dict()}
requests.put('http://127.0.0.1:5000/api/get_scores/2', json = params).json()
```

Appendix 1

Import all libraries for code working:

```
import numpy as np
import pandas as pd
from pandas.core.frame import DataFrame
import os
from flask import abort, Response
from collections import defaultdict
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression, Ridge
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from statsmodels.discrete.discrete_model import Probit
from catboost import CatBoostClassifier, CatBoostRegressor
from sklearn.metrics import roc_auc_score, r2_score, f1_score
from typing import Tuple, Dict, Union, Any
from flask import Flask, request, jsonify, abort, Response
from flask_restx import Api
from modelapi import ML_models
import requests
import pandas as pd
from sklearn.datasets import make_classification, make_regression
from sklearn.model_selection import train_test_split
```

Appendix 2

Requirements:

- catboost==1.2.2

- Flask==3.0.0
- flask_restx==1.2.0
- numpy==1.24.3
- pandas==2.0.2
- Requests==2.31.0
- scikit_learn==1.2.2
- statsmodels==0.14.0
- torch==2.0.1