Branch: master ▾ Find file Copy path

lendingclub-default-predictor / lc-app.py


orangganjil Adjust when pickled objects load

d5c47b7 on Aug 14, 2016

1 contributor

Raw Blame History

84 lines (74 sloc) 3.53 KB

Jump to definition is available! Beta

Navigate your code with ease. In select public repositories, you can now click on function and method calls to jump to their definitions in the same repository. Learn more

Jump to definition is still being calculated for this commit. Check back in a bit. Beta Learn more or give us feedback

```
#!flask/bin/python

from __future__ import division
from flask import Flask, request
from flask_restful import Api, Resource, reqparse
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.externals import joblib

app = Flask(__name__)
api = Api(app)

parser = reqparse.RequestParser()

# Load the pickled label encoder and Random Forest Classifier
le = joblib.load("./lc-label-encoder.pkl")
rfc = joblib.load("./lc-rfc-model.pkl")

class Proba(Resource):
        def post(self):
                json_data = request.get_json()
                df = pd.DataFrame(json_data)
                # Create new DataFrame from nested JSON
                df2 = pd.concat([pd.DataFrame.from_dict(item, orient='index').T for item in df['loans']])
                # Rename some of the columns to match what model expects
                # (LendingClub used different naming conventions for historical data and new data in JSON format)
                df2.rename(columns={'intRate': 'int_rate', 'annualInc': 'annual_inc', 'accNowDelinq': 'acc_now_delinq',
                # Adjust data types for columns we will use
                df2['memberId'] = df2['memberId'].astype(str)
                df2['int_rate'] = df2['int_rate'].astype(int)
                df2['annual_inc'] = df2['annual_inc'].astype(int)
                df2['acc_now_delinq'] = df2['acc_now_delinq'].astype(int)
                df2['term'] = df2['term'].astype(int)
                df2['last_fico_range_high'] = df2['last_fico_range_high'].astype(int)
                df2['last_fico_range_low'] = df2['last_fico_range_low'].astype(int)
                df2['tot_cur_bal'] = df2['tot_cur_bal'].astype(int)
                df2['tot_hi_cred_lim'] = df2['tot_hi_cred_lim'].astype(int)
                # Clean up NaNs and other empty fields (shouldn't be any, but just in case)
                df2['num_tl_30dpd'].fillna(0, inplace=True)
```

```
                df2['int_rate'].fillna(df2['int_rate'].mean(), inplace=True)
                df2['percent_bc_gt_75'].fillna(0, inplace=True)
                df2['dti'].fillna(df2['dti'].mean(), inplace=True)
                # Encode the "term" and "grade" features
                var_mod = ['term','grade']
                for i in var_mod:
                    df2[i] = le.fit_transform(df2[i])
                # List the features to be used in the prediction
                predict_cols = ['int_rate','annual_inc','dti','acc_now_delinq','term','grade','last_fico_range_high','las
                # Create X (predictors)
                X = df2[predict_cols]
                # Make predictions
                y_preds = rfc.predict_proba(X)
                # Retrieve probabilities of default from nested arrays returned by predictor
                proba_defaults = []
                temp_probas = []
                for i in y_preds:
                    temp_probas.append(i[1])
                for x in temp_probas:
                    proba_defaults.append(x)
                # Create new column for probability of default and round to two decimal places
                df2['defaultProb'] = proba_defaults
                df2['defaultProb'] = df2['defaultProb'].round(decimals=2)
                # Create new series consisting of memberId and probability of default columns and return it
                temp_series = df2[['memberId', 'defaultProb']]
                json_out = temp_series.to_json(orient='records')
                return json_out

class Version(Resource):
        def get(self):
                return {'version': '1.0'}

class ReadMe(Resource):
        def get(self):
                return "Submit a list of LendingClub loans in JSON format. A machine learning model will return the membe


api.add_resource(Proba, '/predict')
api.add_resource(Version, '/version')
api.add_resource(ReadMe, '/')

if __name__=='__main__':
        app.run(debug=False)
```