```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun May 17 17:44:18 2020

@author: ankit
"""

from flask import Flask,request #import main Flask Class and request
import pandas as pd
import pickle
import numpy as np
from pandas.io.json import json_normalize

app = Flask(__name__) #create the Flask app


@app.route('/load_model',methods=['POST'])
def load_model():
    req_data = request.get_json()
    test_data_subset = pd.DataFrame.from_dict(json_normalize(req_data),
orient='columns')

    #load the columns to drop file
    columns_to_drop=pd.read_csv("/model/columns_to_drop.csv")

    #select the columns to be retained
    columns_to_Retain = set(test_data_subset.columns.values) -
set(columns_to_drop.colnames.values)
    test_data_selected_columns = test_data_subset[columns_to_Retain]

    #select the categorical columns from the dataframe
    column_datatypes = test_data_selected_columns.dtypes
    categorical_columns =
list(column_datatypes[column_datatypes=="object"].index.values)

    #read the label encoders and apply the encoded values to the categorical
variables
    for cf1 in categorical_columns:
        filename = "/model/"+cf1+".sav"
        le = pickle.load(open(filename, 'rb'))

        #if an new classes is observed, set it to the 0 class
        le_dict = dict(zip(le.classes_, le.transform(le.classes_)))

test_data_selected_columns[cf1]=test_data_selected_columns[cf1].apply(lambda
x: le_dict.get(x, -1))
```

```
    test_data_id = test_data_selected_columns['id']
    test_data_selected_columns = test_data_selected_columns.drop('id',axis=1)

    #convert the interger columns to categories as required by the ML model
    Column_datatypes= test_data_selected_columns.dtypes
    Integer_columns = list(Column_datatypes.where(lambda x: x
=="int64").dropna().index.values)

    #convert the int64 columns categorical
    test_data_selected_columns[Integer_columns] =
test_data_selected_columns[Integer_columns].astype('category',copy=False)

    #load the saved model and predict on the test data
    tuned_model = pickle.load(open("/model/tunedmodel_rf", 'rb'))
    Y_test_predict = tuned_model.predict(test_data_selected_columns)

    #create a new output dataframe
    output = pd.DataFrame()
    output['id']=test_data_id
    output['predict_loss']=Y_test_predict

    output=output.to_json(orient='records')
    return output

if __name__ =='__main__':
    app.run(debug=True,port=4000)#run app in debug mode on port 4000
```