GLM Model

The ipython notebook depicting the training and testing of the dataset was well written, however the code couldn't be readily used to make a testing api.

Previous known Shortcoming:

1. The final model was not being saved for predicting.
2. The scaling and imputation models was also not being saved
3. Pandas' function of dummy variable was hard to save to give the exact same sparse matrix while testing as in training.
4. On deeper understanding of the code it was found while preprocessing dummy variables 3 extra columns, 1 each of x5, x81 and x82 were being generated which had all zeros. This was due to the usage of dummy_na argument which worked fine with x31 but created an extra column for others.
5. The top 25 columns with most significance were also not being saved for later use.

Solutions:

The training file was re-written to overcome shortcoming.

1. Get_dummies was replaced by One Hot Encoder, due to simpler usability and access to save the model.
2. Column names were also saved for later use.
3. All required models such as the Imputer, Standard Scaler and final predicting model were saved.

Folder and file structure:

1. app/test.py : Contains the predicting function. It requires raw data as input in the form of a dictionary or list of dictionaries.

2. app/application.py : Flask API to accept the json data via post request on https://localhost:8080/predict

3. app/models/ : folder containing all the useful models for predicting.
4. app/requirements.txt : This file contains all the required packages for the module
5. ./Dockerfile : Docker file containing all the commands to set up the docker image
6. ./run_api : Shell file which builds the docker image from the dockerfile and run the docker image