In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report
from itertools import product
%matplotlib inline
import pylab
from pylab import rcParams
import statsmodels.api as sm
import statistics
from scipy import stats
import sklearn
import warnings
warnings.filterwarnings('ignore')
import matplotlib as mpl
COLOR = 'white'
mpl.rcParams['text.color'] = COLOR
mpl.rcParams['axes.labelcolor'] = COLOR
mpl.rcParams['xtick.color'] = COLOR
mpl.rcParams['ytick.color'] = COLOR

# Import the dataset to datafram
medical_df = pd.read_csv('C:/Users/MichaelRupert/Downloads/e9d8sm5uf8df75k650df/medical_cleaned_data.csv');

# Rename columns/variables of survey to easily recognizable features (ex: "Item1" to "TimelyResponse").
medical_df.rename(columns = {'Income' : 'Household_Income',
                             'TotalCharge': 'Daily_Average_Charges',
                             'Additional_Charges' : 'Average_Daily_Additional_Services',
                             'Item1':'Timely_Admission','Item2':'Timely_Treatment',
                             'Item3':'Timely_Visits',
                             'Item4':'Reliability','Item5':'Options','Item6':'Hours_Treatment',
                             'Item7':'Courteous_Staff','Item8':'Active_Listening'},inplace = True)
# Get a description of data frame, structure (columns & rows) & data types.
print("_____________________________________________________________")
print("Below is a description and shape of the Data:")
print(medical_df.shape)
print(medical_df.describe())
print("_____________________________________________________________")
#Check for duplicates
Is_dups_bool = medical_df.duplicated()
print("Are there duplicates? ")
print(Is_dups_bool.value_counts())
print("_____________________________________________________________")

medical_df.columns

medical_df['intercept'] = 1
model_VitD = sm.OLS(medical_df['VitD_levels'], medical_df[['State', 'Zip', 'Lat', 'Lng', 'Population', 'Area',
       'Timezone', 'Children', 'Age', 'Education', 'Employment',
       'Household_Income', 'Marital', 'Gender', 'ReAdmis', 'VitD_levels',
       'Doc_visits', 'Full_meals_eaten', 'VitD_supp', 'Soft_drink',
       'Initial_admin', 'HighBlood', 'Stroke', 'Complication_risk',
       'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain',
       'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma',
       'Services', 'Initial_days', 'Daily_Average_Charges',
       'Additional_charges', 'Timely_Admission', 'Timely_Treatment',
       'Timely_Visits', 'Reliability', 'Options', 'Hours_Treatment',
       'Courteous_Staff', 'Active_Listening']]).fit()
#print(lm_ReAdmis.params)
print(model_VitD.summary())
#Remove Unwanted columns
new_Med_DF  =  medical_df.drop(labels=['ReAdmis', 'Soft_drink',
       'Initial_admin',  'Stroke', 'Complication_risk',
       'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain',
       'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma',
       'Services', 'Initial_days', 'Timely_Treatment',
       'Timely_Visits', 'Reliability',  'Active_Listening'],axis=1)
print(new_Med_DF)
print("_____________________________________________________________")

print(new_Med_DF.shape)
print(new_Med_DF.describe())
print("_____________________________________________________________")



# print(new_Med_DF.head(1))
nan_values = new_Med_DF.isnull()
nan_columns = nan_values.any()
columns_with_nan = new_Med_DF.columns[nan_columns].tolist()
print("Is the columns with null values.")
print(columns_with_nan)
print("_____________________________________________________________")

print(new_Med_DF.columns)
```

In [ ]:
```
#Let's Create Two arrays one for Continuous varaibles and one for non-Continuous or Categorical
continuous_Ar = ['Lat', 'Lng', 'Population', 'Children', 'Age',
       'Household_Income',  'VitD_levels', 'VitD_supp', 'Daily_Average_Charges', 'Additional_charges']

categorical_Ar = ['State', 'Zip', 'Area',
       'Timezone', 'Education', 'Employment','Marital', 'Gender',  'Doc_visits',
       'Full_meals_eaten', 'HighBlood', 'Overweight','Timely_Admission',
       'Options', 'Hours_Treatment', 'Courteous_Staff']

full_ar = ['State', 'Zip', 'Lat', 'Lng', 'Population', 'Area',
       'Timezone', 'Children', 'Age', 'Education', 'Employment',
       'Household_Income', 'Marital', 'Gender', 'VitD_levels', 'Doc_visits',
       'Full_meals_eaten', 'VitD_supp', 'HighBlood', 'Overweight',
       'Daily_Average_Charges', 'Additional_charges', 'Timely_Admission',
       'Options', 'Hours_Treatment', 'Courteous_Staff']
#Univariate Distributions
new_Med_DF[['Lat', 'Lng', 'Population', 'Children', 'Age',
       'Household_Income',  'VitD_levels', 'VitD_supp', 'Daily_Average_Charges', 'Additional_charges']].hist()

plt.savefig('medical_pyplot.jpg')
```

In [ ]:
```
#preform boxplots on categorical vairbales

for x in categorical_Ar:

    sns.boxplot(x, data = new_Med_DF)
    plt.show()
```

In [ ]:
```
for s in categorical_Ar:
    sns.catplot(x=s, y="VitD_levels", data=new_Med_DF)
    plt.show()
```

In [ ]:
```
# Run scatterplots to show direct or inverse relationships between target & independent variablesfor x in Varia

for v in (continuous_Ar):
    sns.scatterplot(x=new_Med_DF[v], y=new_Med_DF['VitD_levels'],color='red')
    plt.show();
```

In [ ]:
```
new_Med_DF['intercept'] = 1
model_VitD_clean = sm.OLS(new_Med_DF['VitD_levels'], new_Med_DF[['State', 'Zip', 'Lat',
       'Lng', 'Population', 'Area',
       'Timezone', 'Children', 'Age', 'Education', 'Employment',
       'Household_Income', 'Marital', 'Gender', 'Doc_visits',
       'Full_meals_eaten', 'VitD_supp', 'HighBlood', 'Overweight',
       'Daily_Average_Charges', 'Additional_charges', 'Timely_Admission',
       'Options', 'Hours_Treatment', 'Courteous_Staff']]).fit()
#print(lm_ReAdmis.params)
print(model_VitD_clean.summary())
```

In [ ]:
```
v2_Med_data_cleaned = new_Med_DF.drop(labels=['State','Population','Timezone','Children',
                                              'Age','Employment','Marital', 'VitD_supp','HighBlood',
                                              'Courteous_Staff'],axis=1)

# print(v2_Med_data_cleaned.columns)
# print(v2_Med_data_cleaned.shape)

v2_Med_data_cleaned.to_csv('D208_prepared_Dataset.csv', index=False)
model_V2_Med_Data_cleaned = sm.OLS(new_Med_DF['VitD_levels'], new_Med_DF[['Zip', 'Lat',
       'Lng', 'Area', 'Education',
       'Household_Income', 'Gender',  'Doc_visits',
       'Full_meals_eaten', 'Overweight', 'Daily_Average_Charges',
       'Additional_charges', 'Timely_Admission', 'Options', 'Hours_Treatment']]).fit()
print(model_V2_Med_Data_cleaned.summary())
```

In [ ]:
```
v3_med_data_cleaned = new_Med_DF.drop(columns=[ 'Zip', 'Education', 'Timely_Admission',
                                               'intercept', 'Area','Doc_visits', 'Gender',
                                               'Additional_charges', 'Options', 'Hours_Treatment',
                                               'State','Population','Timezone','Children','Age',
                                               'Employment','Marital', 'VitD_supp','HighBlood',
                                               'Courteous_Staff'],axis=1)
v3_med_data_cleaned.columns
```

In [ ]:
```
model_v3_Med_Data = sm.OLS(v3_med_data_cleaned["VitD_levels"],v3_med_data_cleaned[[ 'Lat', 'Lng',  'Household_I
       'Full_meals_eaten', 'Overweight',
       'Daily_Average_Charges']]).fit()
print(model_v3_Med_Data.summary())
```

In [ ]:
```
print(model_v3_Med_Data.params)
```

In [ ]:
```
mse_mdl_v2 = model_V2_Med_Data_cleaned.mse_resid
mse_mdl_v3 = model_v3_Med_Data.mse_resid
print(np.sqrt(mse_mdl_v2))
print(np.sqrt(mse_mdl_v3))
```

In [ ]:
```
# start exploring the models using imput variables and categorical variables
Lat =  np.arange(0,80, 10)
Lng =  np.arange(-180,-30, 10)
HouseHold_incm = np.arange(0,300000, 10000)
Meals = v3_med_data_cleaned["Full_meals_eaten"].unique()
overweight = v3_med_data_cleaned["Overweight"].unique()
Daily_Average_Charges = np.arange(1000,50000, 2000)

p = product(Lat,Lng,HouseHold_incm,Meals,overweight,Daily_Average_Charges)

explanatory_data = pd.DataFrame(p, columns = [ 'Lat', 'Lng',  'Household_Income',
       'Full_meals_eaten', 'Overweight',
       'Daily_Average_Charges'])
prediction_data_Daily_Charge = explanatory_data.assign(VitD_levels = model_v3_Med_Data.predict(explanatory_data
# #sns.scatterplot(x="Daily_Average_Charges", y="State",)
print(prediction_data_Daily_Charge)
```

In [ ]:
```
# start exploring the models using imput variables and categorical variables

Lat =  np.arange(0,80, 10)
Lng =  np.arange(-180,-30, 10)
HouseHold_incm = np.arange(0,300000, 10000)
VitD = np.arange(5,80, 5)
Meals = v3_med_data_cleaned["Full_meals_eaten"].unique()
overweight = v3_med_data_cleaned["Overweight"].unique()
Daily_Average_Charges = np.arange(1000,50000, 2000)

p = product(Lat, Lng, HouseHold_incm , Meals, overweight, Daily_Average_Charges)

explanatory_data = pd.DataFrame(p, columns = ['Lat', 'Lng', 'Household_Income', 'Full_meals_eaten', 'Overweight
```

In [ ]:
```
prediction_data_Daily_Charge= explanatory_data.assign(VitD_Levels = model_v4_Med_Data.predict(explanatory_data)

print(prediction_data_Daily_Charge)
```

In [ ]:
```

```