In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import OrdinalEncoder
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import mean_absolute_error as mae
from sklearn.metrics import roc_auc_score

path='/Users/herlihpj/Desktop/Data Analytics/D209 - Data Mining/Task 2/'

#Data Preparation
#Reads CSV to data frame, sets case order to index
med_dirty= pd.read_csv(path+'medical_clean.csv',
                         index_col=0)
#Check for Null
print('Summary of Null: ')
print(med_dirty.isna().sum())

#Check for duplicated data
duplicates=med_dirty.duplicated()
print('Duplicates: ;', duplicates.sum())

# Dropping columns not relavant to the analysis
med_mine = med_dirty.drop(columns= ["Customer_id", "Interaction",'TimeZone', "UID", "City", "County", "Zip", "Lat",
                            "Lng",'Job','Item1','Item2', 'Item3','Item4','Item5','Item6','Item7','Item8'])

#Explore the data/statistics
print(med_mine.head())
print(med_mine.describe())
print(med_mine.info())

#Various Scatterplots to Visualize data
sns.scatterplot(data=med_mine, x="Initial_days", y="TotalCharge", hue='Initial_admin')
plt.show()
sns.scatterplot(data=med_mine, x="Initial_days", y="VitD_levels", hue='ReAdmis')
plt.show()

#Ordinal Encoding to convert to numeric 0:No, 1:Yes; other variable alphabetically starting with 0
oe_dict={}
#list of columns to convert to numerical
```

```
convert_cols=['State','Area','Services', 'Marital', 'Gender','Soft_drink', 'Initial_admin', 'HighBlood', 'Stroke',
              'Complication_risk','Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia','BackPain', 'Anxiety',
             'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma', 'Services','ReAdmis']#
for col_name in convert_cols:
    #print(col_name+' pre: '+str(med_mine[col_name].unique()))
    #Creates column ordinal encoder
    oe_dict[col_name]=OrdinalEncoder()
    col=med_mine[col_name]
    #select non-null values of col
    col_not_null=col[col.notnull()]
    reshaped_vals=col_not_null.values.reshape(-1,1)
    encoded_vals=oe_dict[col_name].fit_transform(reshaped_vals)
    med_mine.loc[col.notnull(), col_name]=np.squeeze(encoded_vals)
    #print(col_name+' post: '+str(med_mine[col_name].unique()))

#Visual EDA - wouldnt plot with color y?
#_= pd.plotting.scatter_matrix(med_mine, c = 'green', figsize = [8, 8],s=150, marker = 'D')
#returns a series of plots and histograms
#Slow runtime too many graphs with this data

med_mine.to_csv(path+'Prepared_data.csv')
print('Prepared Data has been exported to CSV')

print('====================== \n Data has been prepared     \n====================== ')

#Function which takes y_test values, prediction values, and a range to compare against
#Function scores predictions based on accuracy
def score_within(test, predictions, buffer):
    score=0
    for x, pred in enumerate(predictions):
        test_min=test.iloc[x]-buffer
        test_max=test.iloc[x]+buffer
        #print('Correct: ',y_test.iloc[x], ' Predicted: ', pred)
        if pred>test_min and pred<test_max:
            score=score+1
    return score

### RANDOM FOREST - Hyperparameter Tuned REGRESSION with Grid Search Cross Validation ###
print('RANDOM FORESTS:')

#Set the target variable
target='Initial_days'

#Drop additional columns not necessary to the analysis
med_mine=med_mine.drop(columns = ['Additional_charges'])#'TotalCharge','ReAdmis',
#Train/test Split
X=med_mine.drop(target, axis=1)#.values #can drop readmis too doesnt change anything
y=med_mine[target]
#Check to make sure target has same numer of rows and just one value
print(X.shape) #(10000, 27)
print(y.shape) #(10000,)

# Set seed for reproducibility
SEED = 1
# Split dataset into 70% train and 30% test
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.3,random_state=SEED)

#Export split data
# Convert X_train and y_train into DataFrames
X_train_df = pd.DataFrame(X_train)
y_train_df = pd.DataFrame(y_train)
X_test_df = pd.DataFrame(X_test)
y_test_df = pd.DataFrame(y_test)
# Merge X_train_df and y_train_df into a single DataFrame
train = X_train_df.join(y_train_df)
test = X_test_df.join(y_test_df)
#Export to CSV's
train.to_csv(path+'Training_data.csv')
test.to_csv(path+'Testing_data.csv')
print('Split data has been exported')

# Instantiate a random forests regressor 'rf'
rf = RandomForestRegressor(random_state= SEED)
# Inspect rf' s hyperparameters
print(rf.get_params())

params_rf = {'n_estimators': [100, 200, 300],
             'max_depth': [4, 6, 8],
             'min_samples_leaf': [0.08, 0.1,0.15],
             'max_features': ['log2','sqrt']}

# Instantiate 'grid_rf'
grid_rf = GridSearchCV(estimator=rf,param_grid=params_rf,cv=3, scoring='neg_mean_squared_error',verbose=1,n_jobs=-1)

#Search for the Best Hyper Parameters
# Fit 'grid_rf' to the training set
grid_rf.fit(X_train, y_train)
#output shows messages to grid fitting and the obtained optimal model
```

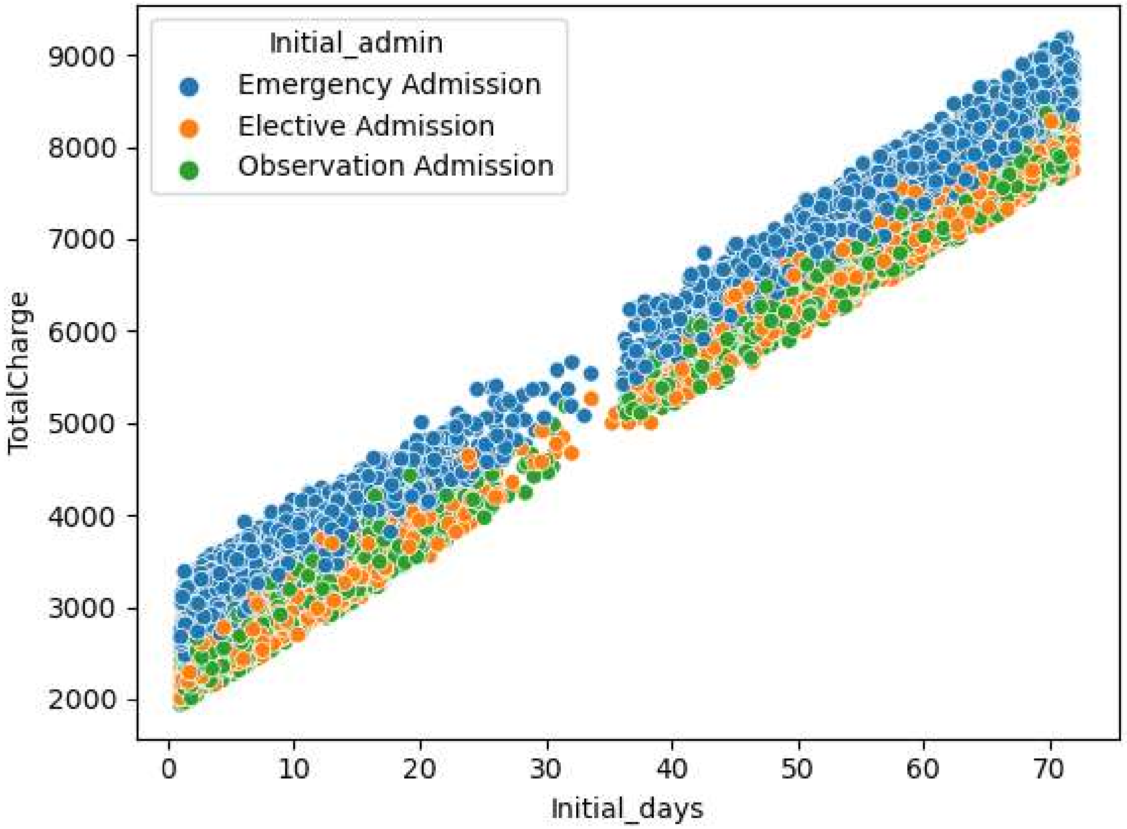
```
# Extract the best hyperparameters from 'grid_rf'
best_hyperparams = grid_rf.best_params_
print('Best hyperparameters:\n', best_hyperparams)

# Extract the best model from 'grid_rf'
best_model = grid_rf.best_estimator_
# Predict the test set labels
y_pred = best_model.predict(X_test)

# Feature Importance
# Create a pd.Series of features importances
importances_rf = pd.Series(data=best_model.feature_importances_,index=X.columns)
# Sort importances_rf
sorted_importances_rf = importances_rf.sort_values()
# Make a horizontal bar plot
sorted_importances_rf.plot(kind='barh', color='lightgreen'); plt.show()
#Plots a horizontal bar graph with each features importance

#Can also store and sort the importances first then loop through
#print('Top Features Causing '+target+':')
#for i, item in enumerate(sorted_importances_rf):
#    if (item>.05):
#        print("{0:s}: {1:.2f}".format(X.columns[i], item))

days_within=14
print('Score: ', score_within(y_test, y_pred, days_within), ' Of: ', len(y_pred))

# Evaluate the train & test set MAE
##
print('Train vs Test MAEs:')
# Create vectors of predictions
train_predictions = best_model.predict(X_train)
test_predictions = best_model.predict(X_test)
# Train/Test Errors
train_error = mae(y_true=y_train, y_pred=train_predictions)
test_error = mae(y_true=y_test, y_pred=test_predictions)
# Print the accuracy for seen and unseen data
print("Model error on seen data: {0:.2f}.".format(train_error))
print("Model error on unseen data: {0:.2f}.".format(test_error))
##

# Evaluate the train & test set MSE and RMSE
mse_test=MSE(y_test, y_pred)
rmse_test = mse_test**(1/2)
# Print the test set MSE
```

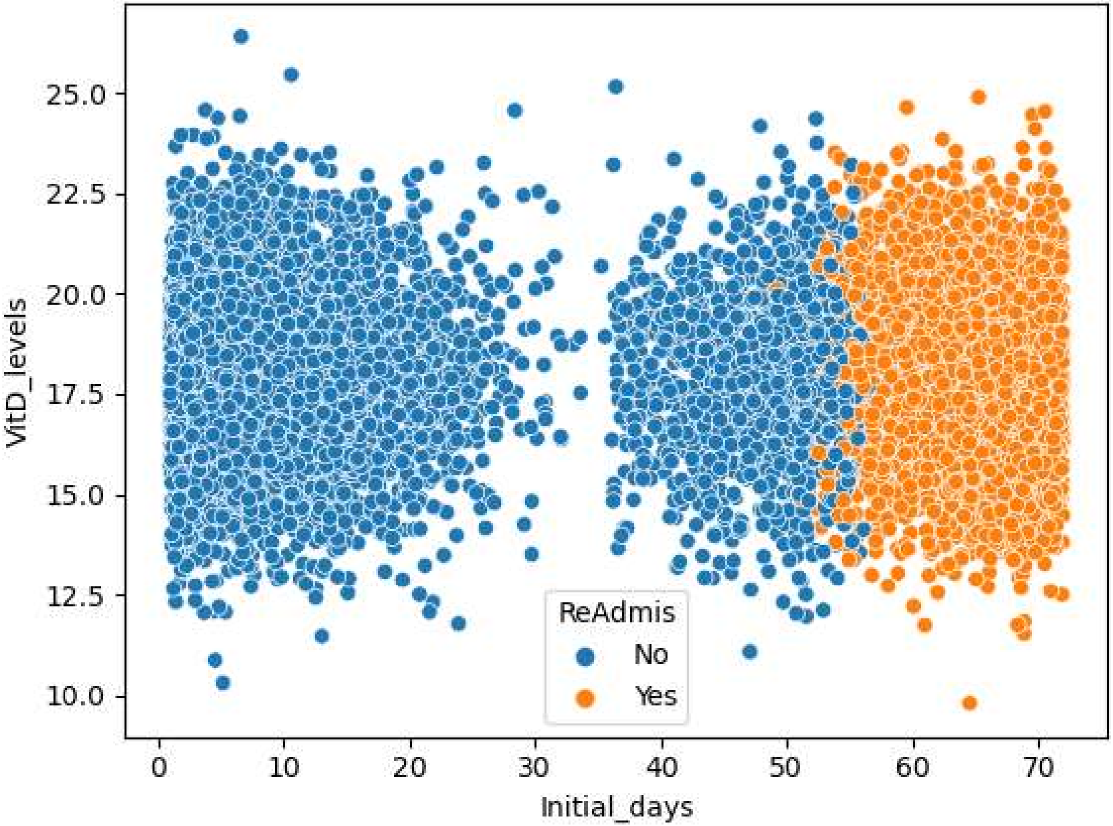
```
print('Test set MSE: {:.2f}'.format(mse_test))
# Print the test set RMSE
print('Test set RMSE: {:.2f}'.format(rmse_test))
#Test set
print("Test Set MAE: {0:.2f}".format(mae(y_true=y_test, y_pred=y_pred)))
```

```
Summary of Null:
Customer_id           0
Interaction           0
UID                   0
City                  0
State                 0
County                0
Zip                   0
Lat                   0
Lng                   0
Population            0
Area                  0
TimeZone              0
Job                   0
Children              0
Age                   0
Income                0
Marital               0
Gender                0
ReAdmis               0
VitD_levels           0
Doc_visits            0
Full_meals_eaten      0
vitD_supp             0
Soft_drink            0
Initial_admin         0
HighBlood             0
Stroke                0
Complication_risk     0
Overweight            0
Arthritis             0
Diabetes              0
Hyperlipidemia        0
BackPain              0
Anxiety               0
Allergic_rhinitis     0
Reflux_esophagitis    0
Asthma                0
Services              0
Initial_days          0
TotalCharge           0
Additional_charges    0
Item1                 0
Item2                 0
Item3                 0
```

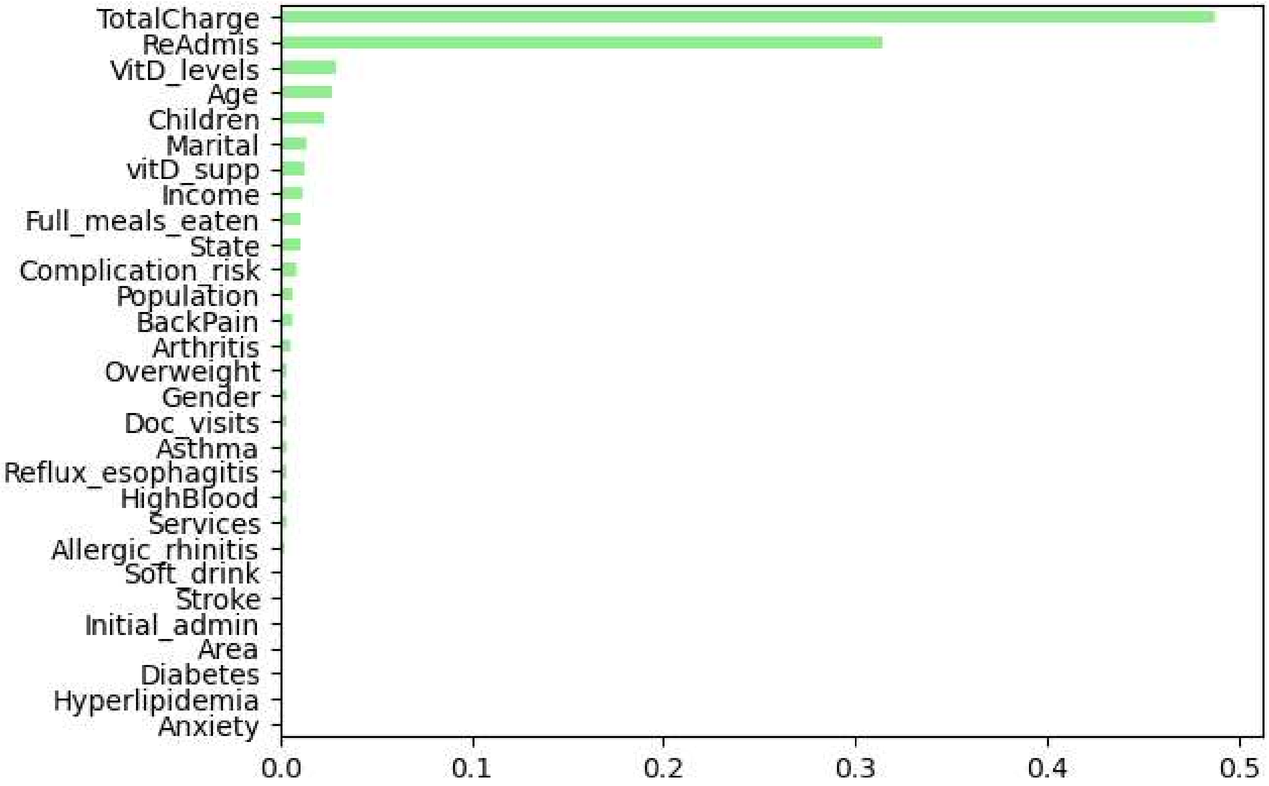
```
Item4                 0
Item5                 0
Item6                 0
Item7                 0
Item8                 0
dtype: int64
Duplicates: ; 0
          State  Population      Area  Children  Age    Income   Marital  \
CaseOrder                                                                  
1            AL        2951  Suburban         1   53  86575.93  Divorced   
2            FL       11303     Urban         3   51  46805.99   Married   
3            SD       17125  Suburban         3   53  14370.14   Widowed   
4            MN        2162  Suburban         0   78  39741.49   Married   
5            VA        5287     Rural         1   22   1209.56   Widowed   

           Gender ReAdmis  VitD_levels  ...  Hyperlipidemia  BackPain  \
CaseOrder                               ...                             
1            Male      No    19.141466  ...              No       Yes   
2          Female      No    18.940352  ...              No        No   
3          Female      No    18.057507  ...              No        No   
4            Male      No    16.576858  ...              No        No   
5          Female      No    17.439069  ...             Yes        No   

           Anxiety Allergic_rhinitis Reflux_esophagitis Asthma     Services  \
CaseOrder                                                                     
1              Yes               Yes                 No    Yes   Blood Work   
2               No                No                Yes     No  Intravenous   
3               No                No                 No     No   Blood Work   
4               No                No                Yes    Yes   Blood Work   
5               No               Yes                 No     No      CT Scan   

          Initial_days  TotalCharge Additional_charges  
CaseOrder                                               
1            10.585770  3726.702860       17939.403420  
2            15.129562  4193.190458       17612.998120  
3             4.772177  2434.234222       17505.192460  
4             1.714879  2127.830423       12993.437350  
5             1.254807  2113.073274        3716.525786  

[5 rows x 31 columns]
          Population      Children           Age         Income   VitD_levels  \
count   10000.000000  10000.000000  10000.000000   10000.000000  10000.000000   
mean     9965.253800      2.097200     53.511700   40490.495160     17.964262   
std     14824.758614      2.163659     20.638538   28521.153293      2.017231   
min         0.000000      0.000000     18.000000     154.080000      9.806483   
```

```
25%       694.750000      0.000000     36.000000   19598.775000     16.626439
50%      2769.000000      1.000000     53.000000   33768.420000     17.951122
75%     13945.000000      3.000000     71.000000   54296.402500     19.347963
max    122814.000000     10.000000     89.000000  207249.100000     26.394449

         Doc_visits  Full_meals_eaten     vitD_supp  Initial_days  \
count  10000.000000      10000.000000  10000.000000  10000.000000   
mean       5.012200          1.001400      0.398900     34.455299   
std        1.045734          1.008117      0.628505     26.309341   
min        1.000000          0.000000      0.000000      1.001981   
25%        4.000000          0.000000      0.000000      7.896215   
50%        5.000000          1.000000      0.000000     35.836244   
75%        6.000000          2.000000      1.000000     61.161020   
max        9.000000          7.000000      5.000000     71.981490   

        TotalCharge  Additional_charges  
count  10000.000000        10000.000000  
mean    5312.172769        12934.528587  
std     2180.393838         6542.601544  
min     1938.312067         3125.703000  
25%     3179.374015         7986.487755  
50%     5213.952000        11573.977735  
75%     7459.699750        15626.490000  
max     9180.728000        30566.070000  
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10000 entries, 1 to 10000
Data columns (total 31 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   State               10000 non-null  object 
 1   Population          10000 non-null  int64  
 2   Area                10000 non-null  object 
 3   Children            10000 non-null  int64  
 4   Age                 10000 non-null  int64  
 5   Income              10000 non-null  float64
 6   Marital             10000 non-null  object 
 7   Gender              10000 non-null  object 
 8   ReAdmis             10000 non-null  object 
 9   VitD_levels         10000 non-null  float64
 10  Doc_visits          10000 non-null  int64  
 11  Full_meals_eaten    10000 non-null  int64  
 12  vitD_supp           10000 non-null  int64  
 13  Soft_drink          10000 non-null  object 
 14  Initial_admin       10000 non-null  object 
 15  HighBlood           10000 non-null  object 
 16  Stroke              10000 non-null  object 
 17  Complication_risk   10000 non-null  object 
 18  Overweight          10000 non-null  object 
 19  Arthritis           10000 non-null  object 
 20  Diabetes            10000 non-null  object 
 21  Hyperlipidemia      10000 non-null  object 
 22  BackPain            10000 non-null  object 
 23  Anxiety             10000 non-null  object 
 24  Allergic_rhinitis   10000 non-null  object 
 25  Reflux_esophagitis  10000 non-null  object 
 26  Asthma              10000 non-null  object 
 27  Services            10000 non-null  object 
 28  Initial_days        10000 non-null  float64
 29  TotalCharge         10000 non-null  float64
 30  Additional_charges  10000 non-null  float64
dtypes: float64(5), int64(6), object(20)
memory usage: 2.4+ MB
None
```





```
Prepared Data has been exported to CSV
=======================
 Data has been prepared
=======================
RANDOM FORESTS:
(10000, 29)
(10000,)
Split data has been exported
{'bootstrap': True, 'ccp_alpha': 0.0, 'criterion': 'squared_error', 'max_depth': None, 'max_features': 'auto', 'max_leaf
_nodes': None, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_we
ight_fraction_leaf': 0.0, 'n_estimators': 100, 'n_jobs': None, 'oob_score': False, 'random_state': 1, 'verbose': 0, 'war
m_start': False}
Fitting 3 folds for each of 54 candidates, totalling 162 fits
Best hyperparameters:
 {'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 0.08, 'n_estimators': 300}
```



```
Score:  1955  Of:  3000
Train vs Test MAEs:
Model error on seen data: 11.31.
Model error on unseen data: 11.39.
Test set MSE: 155.91
Test set RMSE: 12.49
Test Set MAE: 11.39
```

```
In [ ]:
```