In [1]:
```
import numpy as np
import pandas as pd
from sklearn import linear_model
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import sklearn
from sklearn import datasets
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report
pd.set_option('display.max_columns', None)
df = pd.read_csv (r'C:\Users\fahim\Documents\A_WGUDocuments\d209\medical_clean.csv')
df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   CaseOrder           10000 non-null  int64
 1   Customer_id         10000 non-null  object
 2   Interaction         10000 non-null  object
 3   UID                 10000 non-null  object
 4   City                10000 non-null  object
 5   State               10000 non-null  object
 6   County              10000 non-null  object
 7   Zip                 10000 non-null  int64
 8   Lat                 10000 non-null  float64
 9   Lng                 10000 non-null  float64
 10  Population          10000 non-null  int64
 11  Area                10000 non-null  object
 12  TimeZone            10000 non-null  object
 13  Job                 10000 non-null  object
 14  Children            10000 non-null  int64
 15  Age                 10000 non-null  int64
 16  Income              10000 non-null  float64
 17  Marital             10000 non-null  object
 18  Gender              10000 non-null  object
 19  ReAdmis             10000 non-null  object
 20  VitD_levels         10000 non-null  float64
 21  Doc_visits          10000 non-null  int64
 22  Full_meals_eaten    10000 non-null  int64
 23  vitD_supp           10000 non-null  int64
 24  Soft_drink          10000 non-null  object
 25  Initial_admin       10000 non-null  object
 26  HighBlood           10000 non-null  object
 27  Stroke              10000 non-null  object
 28  Complication_risk   10000 non-null  object
 29  Overweight          10000 non-null  object
 30  Arthritis           10000 non-null  object
 31  Diabetes            10000 non-null  object
 32  Hyperlipidemia      10000 non-null  object
 33  BackPain            10000 non-null  object
 34  Anxiety             10000 non-null  object
 35  Allergic_rhinitis   10000 non-null  object
 36  Reflux_esophagitis  10000 non-null  object
 37  Asthma              10000 non-null  object
 38  Services            10000 non-null  object
 39  Initial_days        10000 non-null  float64
 40  TotalCharge         10000 non-null  float64
 41  Additional_charges  10000 non-null  float64
 42  Item1               10000 non-null  int64
 43  Item2               10000 non-null  int64
 44  Item3               10000 non-null  int64
 45  Item4               10000 non-null  int64
 46  Item5               10000 non-null  int64
 47  Item6               10000 non-null  int64
 48  Item7               10000 non-null  int64
 49  Item8               10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

In [3]:
```
#check for missing data entries
df.isnull().any()
```

Out[3]:
```
CaseOrder             False
Customer_id           False
Interaction           False
UID                   False
City                  False
State                 False
County                False
Zip                   False
Lat                   False
Lng                   False
Population            False
Area                  False
TimeZone              False
Job                   False
Children              False
Age                   False
Income                False
Marital               False
Gender                False
ReAdmis               False
VitD_levels           False
Doc_visits            False
Full_meals_eaten      False
vitD_supp             False
Soft_drink            False
Initial_admin         False
HighBlood             False
Stroke                False
Complication_risk     False
Overweight            False
Arthritis             False
Diabetes              False
Hyperlipidemia        False
BackPain              False
Anxiety               False
Allergic_rhinitis     False
Reflux_esophagitis    False
Asthma                False
Services              False
Initial_days          False
TotalCharge           False
Additional_charges    False
Item1                 False
Item2                 False
Item3                 False
Item4                 False
Item5                 False
Item6                 False
Item7                 False
Item8                 False
dtype: bool
```

In [4]:
```
#check for any duplicate data entries in columns
df[df.duplicated()]
```

Out[4]:

CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	Population	Area	TimeZone	Job	Children	Age	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	TotalCharge	Additional_charges	Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8

In [5]:
```
#check if any columns are duplicated - looking for false
df.columns.duplicated().any()
```

Out[5]:
```
False
```

In [6]:
```
#check if any rows are duplicated - looking for false
df.duplicated().any()
```

Out[6]:
```
False
```

In [7]:
```
# drop demographic data
df = df.drop(["CaseOrder','Customer_id','Interaction','UID','City','State','County','Zip','Lat','Lng','Population','Area','TimeZone','Job'], axis=1)
```

In [8]:
```
# confirm that columns were dropped
df.head()
```

Out[8]:

Children	Age	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	TotalCharge	Additional_charges	Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8
0	1	53	86575.93	Divorced	Male	No	19:14:1466	6	0	0	No	Emergency Admission	Yes	No	Medium	No	Yes	Yes	No	Yes	Yes	No	Yes	Blood Work	10.585770	3735.702060	17939.403420	3	4	3	3	4	3	3	4
1	3	51	46805.99	Married	Female	No	19:40:6352	4	2	1	No	Emergency Admission	Yes	No	High	Yes	No	No	No	No	No	No	No	Intravenous	15.129562	4183.190458	17612.998120	3	4	3	3	4	3	3	4
2	3	53	14370.14	Widowed	Female	No	18:057507	4	1	0	No	Elective Admission	Yes	No	Medium	Yes	No	Yes	No	No	No	No	No	Blood Work	4.772177	2434.234222	17505.192440	3	5	3	3	4	3	3	4
3	0	78	39741.49	Married	Male	No	16:578058	4	1	0	No	Elective Admission	No	No	Medium	No	Yes	No	No	No	No	No	Yes	Intravenous	1.714878	2127.830423	12965.437350	3	4	2	2	4	3	3	3
4	1	22	1209.56	Widowed	Female	No	17:439069	5	0	2	Yes	Elective Admission	No	No	Low	No	No	Yes	No	Yes	No	No	No	Blood Work	1.254087	2113.073274	3716.525786	2	4	2	2	4	2	3	4

In [9]:
```
#overview of descriptive statistics
df.describe()
```

Out[9]:

	Children	Age	Income	VitD_levels	Doc_visits	Full_meals_eaten	vitD_supp	Soft_drink	Initial_days	TotalCharge	Additional_charges	Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	2.097200	53.511700	40490.495160	17.964262	5.012200	1.001400	0.398900	34.455299	5312.172769	12954.528587	3.516800	3.506700	3.511100	3.515100	3.495900	3.522500	3.484000	3.509700	
std	2.163659	20.638538	28521.153293	2.017231	1.045734	1.008117	0.629506	26.309341	2180.393838	6542.601544	1.031966	1.034625	1.032755	1.036282	1.030192	1.032376	1.021405	1.042312	
min	0.000000	18.000000	154.080000	9.806483	1.000000	0.000000	0.000000	1.001981	1838.352927	3125.703000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	
25%	0.000000	36.000000	19586.775000	16.626439	4.000000	0.000000	0.000000	7.896215	3179.374015	7986.487755	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	
50%	1.000000	53.000000	33768.420000	17.951122	5.000000	1.000000	0.000000	35.836244	5213.952000	11573.977735	4.000000	3.000000	4.000000	4.000000	3.000000	4.000000	3.000000	3.000000	
75%	3.000000	71.000000	54296.402500	19.347963	6.000000	2.000000	1.000000	61.161020	7459.690750	16626.404000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	
max	10.000000	89.000000	207249.100000	26.394449	9.000000	7.000000	5.000000	71.981490	9180.728000	30566.070000	8.000000	7.000000	8.000000	7.000000	7.000000	7.000000	7.000000	7.000000	

In [10]:
```
#rename survey columns for easier identification
df.rename(columns={'Item1':'Timely_admin','Item2':'Timely_treat','Item3':'Timely_visits','Item4':'Reliability','Item5':'Options','Item6':'Hrs_treat','Item7':'Courteous','Item8':'Active_listen'},inplace=True)
```

In [11]:
```
#verify that the survey columns were renamed correctly
df.head()
```

Out[11]:

Children	Age	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	TotalCharge	Additional_charges	Timely_admin	Timely_treat	Timely_visits	Reliability	Options	Hrs_treat	Courteous	Active_listen
0	1	53	86575.93	Divorced	Male	No	19:14:1466	6	0	0	No	Emergency Admission	Yes	No	Medium	No	Yes	Yes	No	Yes	Yes	No	Yes	Blood Work	10.585770	3735.702060	17939.403420	3	4	3	3	4	3	3	4
1	3	51	46805.99	Married	Female	No	19:40:6352	4	2	1	No	Emergency Admission	Yes	No	High	Yes	No	No	No	No	No	No	No	Intravenous	15.129562	4183.190458	17612.998120	3	4	3	3	4	3	3	4
2	3	53	14370.14	Widowed	Female	No	18:057507	4	1	0	No	Elective Admission	Yes	No	Medium	Yes	No	Yes	No	No	No	No	No	Blood Work	4.772177	2434.234222	17505.192440	3	5	3	3	4	3	3	4
3	0	78	39741.49	Married	Male	No	16:578058	4	1	0	No	Elective Admission	No	No	Medium	No	Yes	No	No	No	No	No	Yes	Intravenous	1.714878	2127.830423	12965.437350	3	4	2	2	4	3	3	3
4	1	22	1209.56	Widowed	Female	No	17:439069	5	0	2	Yes	Elective Admission	No	No	Low	No	No	Yes	No	Yes	No	No	No	Blood Work	1.254087	2113.073274	3716.525786	2	4	2	2	4	2	3	4

In [12]:
```
#change yes/no to 1/0
df = df.replace(to_replace = ['Yes','No'],value = [1,0])
```

In [13]:
```
#verify that the values were changed
df.head()
```

Out[13]:

Children	Age	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	TotalCharge	Additional_charges	Timely_admin	Timely_treat	Timely_visits	Reliability	Options	Hrs_treat	Courteous	Active_listen
0	1	53	86575.93	Divorced	Male	0	19:14:1466	6	0	0	0	Emergency Admission	1	0	Medium	0	1	1	0	1	1	0	1	Blood Work	10.585770	3735.702060	17939.403420	3	4	3	3	4	3	3	4
1	3	51	46805.99	Married	Female	0	19:40:6352	4	2	1	0	Emergency Admission	1	0	High	1	0	0	0	0	0	0	0	Intravenous	15.129562	4183.190458	17612.998120	3	4	3	3	4	3	3	4
2	3	53	14370.14	Widowed	Female	0	18:057507	4	1	0	0	Elective Admission	1	0	Medium	1	0	1	0	0	0	0	0	Blood Work	4.772177	2434.234222	17505.192440	3	5	3	3	4	3	3	4
3	0	78	39741.49	Married	Male	0	16:578058	4	1	0	0	Elective Admission	0	1	Medium	0	1	0	0	0	0	0	1	Intravenous	1.714878	2127.830423	12965.437350	3	4	2	2	4	3	3	3
4	1	22	1209.56	Widowed	Female	0	17:439069	5	0	2	1	Elective Admission	0	0	Low	0	0	1	0	1	0	0	0	Blood Work	1.254087	2113.073274	3716.525786	2	4	2	2	4	2	3	4

In [14]:
```
# drop non-numeric variables that are less relevant
df = df.drop(['Marital', 'Gender', 'Initial_admin', 'Complication_risk', 'Services'], axis=1)
```

In [15]:
```
#verify that the non-numeric variables were dropped
df.head()
```

Out[15]:

Children	Age	Income	ReAdmis	VitD_levels	Doc_visits	Full_meals_eaten	vitD_supp	Soft_drink	HighBlood	Stroke	Overweight	Arthritis	Diabetes	Hyperlipidemia	BackPain	Anxiety	Allergic_rhinitis	Reflux_esophagitis	Asthma	Initial_days	TotalCharge	Additional_charges	Timely_admin	Timely_treat	Timely_visits	Reliability	Options	Hrs_treat	Courteous	Active_listen
0	1	53	86575.93	0	19:14:1466	6	0	0	0	1	0	0	1	1	0	1	1	1	0	1	10.585770	3735.702060	17939.403420	3	4	3	3	4	3	3	4
1	3	51	46805.99	0	19:40:6352	4	2	1	0	1	0	1	0	0	0	0	0	0	1	0	15.129562	4183.190458	17612.998120	3	4	3	3	4	3	3	4
2	3	53	14370.14	0	18:057507	4	1	0	0	1	0	1	0	1	0	0	0	0	0	0	4.772177	2434.234222	17505.192440	3	5	3	3	4	3	3	4
3	0	78	39741.49	0	16:578058	4	1	0	0	0	1	0	1	0	0	0	0	0	1	1	1.714878	2127.830423	12965.437350	3	4	2	2	4	3	3	3
4	1	22	1209.56	0	17:439069	5	0	2	1	0	0	0	0	1	0	1	0	0	0	0	1.254087	2113.073274	3716.525786	2	4	2	2	4	2	3	4

In [16]:
```
df.to_csv('C:\Users\fahim\Documents\A_WGUDocuments\d209\medical_D209TASK1PREPARED.csv', index=False)
```

In [17]:
```
#set predictor variables and target variable
X = df.drop('ReAdmis', axis=1).values
y=df['ReAdmis'].values
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score, train_test_split
```

In [18]:
```
#set seed in order to reproduce
SEED=1
```

In [19]:
```
#create training and test datasets
X_train,X_test,Y_train,Y_test = train_test_split(X,y,test_size=0.20,random_state=SEED)
```

In [20]:
```
#export test and training data
X_train.tofile('C:\Users\fahim\Documents\A_WGUDocuments\d209\medical_Xtrain.csv',sep=',')
X_test.tofile('C:\Users\fahim\Documents\A_WGUDocuments\d209\medical_Xtest.csv',sep=',')
Y_train.tofile('C:\Users\fahim\Documents\A_WGUDocuments\d209\medical_Ytrain.csv',sep=',')
Y_test.tofile('C:\Users\fahim\Documents\A_WGUDocuments\d209\medical_Ytest.csv',sep=',')
```

In [21]:
```
#begin the KNN model
knn=KNeighborsClassifier(n_neighbors=7)
```

In [22]:
```
#fit data to KNN model
knn.fit(X_train,Y_train)
```

Out[22]:
```
KNeighborsClassifier(n_neighbors=7)
```

In [23]:
```
#predict outcomes from test data
Y_pred=knn.predict(X_test)
```

In [24]:
```
#show initial accuracy score of KNN model
print('Initial accuracy score of KNN model: ',accuracy_score(Y_test,Y_pred))
```

```
Initial accuracy score of KNN model:  0.921
```

In [25]:
```
#compute classification metrics
print(classification_report(Y_test,Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.94      1261
           1       0.85      0.65      0.90       739

    accuracy                           0.92      2000
   macro avg       0.91      0.93      0.92      2000
weighted avg       0.93      0.92      0.92      2000
```

In [26]:
```
#scale data
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
steps=[('scaler',StandardScaler()),('KNN',KNeighborsClassifier())]
pipeline=Pipeline(steps)
```

In [27]:
```
#split data
X_train_scaled,X_test_scaled,Y_train_scaled,Y_test_scaled=train_test_split(x,y,test_size=0.20,random_state=SEED)
```

In [28]:
```
#scale data with pipeline
KNN_scaled=pipeline.fit(X_train_scaled,Y_train_scaled)
```

In [29]:
```
#predict from scaled data
Y_pred_scaled=pipeline.predict(X_test_scaled)
```

In [30]:
```
#show new accuracy score of scaled KNN model
print('New accuracy score of scaled KNN model:{:.3f}'.format(accuracy_score(Y_test_scaled,Y_pred_scaled)))
```

```
New accuracy score of scaled KNN model:0.890
```

In [31]:
```
#ccompute new classification metrics after scaling
print(classification_report(Y_test_scaled,Y_pred_scaled))
```

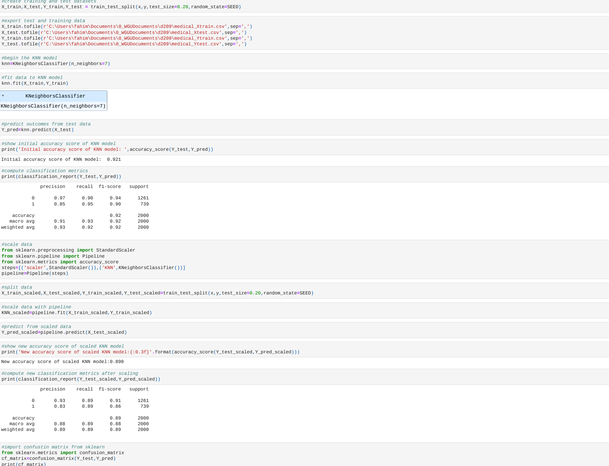
```
              precision    recall  f1-score   support

           0       0.93      0.89      0.91      1261
           1       0.83      0.89      0.86       739

    accuracy                           0.89      2000
   macro avg       0.88      0.89      0.88      2000
weighted avg       0.89      0.89      0.89      2000
```

In [32]:
```
#import confusion matrix from sklearn
from sklearn.metrics import confusion_matrix
cf_matrix=confusion_matrix(Y_test,Y_pred)
print(cf_matrix)
```

```
[[1140  121]
 [  37  702]]
```

In [33]:
```
#visualize confusion matrix
group_names = ["True Neg","False Pos","False Neg","True Pos"]
group_counts=["{0:0.0f}".format(value) for value in cf_matrix.flatten()]
group_percentages=["{0:.2%}".format(value) for value in cf_matrix.flatten()/np.sum(cf_matrix)]
labels=["{v1}\n{v2}\n{v3}".format(v1,v2,v3) for v1,v2,v3 in
zip(group_names,group_counts,group_percentages)]
labels=np.asarray(labels).reshape(2,2)
sns.heatmap(cf_matrix,annot=labels,fmt='',cmap='Greens')
```

Out[33]:
```
<AxesSubplot:>
```



In [34]:
```
#import GridSearch for cross validation
from sklearn.model_selection import GridSearchCV
```

In [35]:
```
#set parameters
param_grid={'n_neighbors':np.arange(1,50)}
```

In [36]:
```
#recall KNN for cross validation
knn=KNeighborsClassifier()
```

In [37]:
```
#begin GridSearch cross validation
knn_cv=GridSearchCV(KNN,param_grid,cv=5)
```

In [38]:
```
#fit model
knn_cv.fit(X_train,y_train)
```

Out[38]:
```
GridSearchCV
estimator: KNeighborsClassifier
KNeighborsClassifier
```

In [39]:
```
#show best parameters
print('Best parameters for the KNN model: {}'.format(knn_cv.best_params_))
```

```
Best parameters for the KNN model: {'n_neighbors': 3}
```

In [40]:
```
#show model best score
print('Best score for the KNN model: {:.3f}'.format(knn_cv.best_score_))
```

```
Best score for the KNN model: 0.918
```

In [41]:
```
#import ROC AUC to explain area under curve
from sklearn.metrics import roc_auc_score
```

In [42]:
```
#fit to data
knn_cv.fit(x,y)
```

Out[42]:
```
GridSearchCV
estimator: KNeighborsClassifier
KNeighborsClassifier
```

In [44]:
```
#determine predicted probabilities
Y_pred_prob=knn_cv.predict_proba(X_test)[:,1]
```

In [45]:
```
#determine and show AUC score
print("The Area Under Curve (AUC) on validation data is:{:.4f}".format(roc_auc_score(Y_test,Y_pred_prob)))
```

```
The Area Under Curve (AUC) on validation data is:0.9765
```

In [46]:
```
#determine cross-validated AUC scores
cv_auc=cross_val_score(knn_cv,x,y,cv=5,scoring='roc_auc')
```

In [48]:
```
#show AUC scores
print("AUC scores computed using 5-fold cross validation: {}".format(cv_auc))
```

```
AUC scores computed using 5-fold cross validation: [0.94054771 0.9372297  0.94170315 0.92813728 0.9558256 ]
```

In [ ]:
```

```