In [4]:

```
#import packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [5]:

```
# Load data
data = pd.read_csv("diabetes.csv")

# check header data
data.head()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2

Project Task: Week 1

Data Exploration:

1. Perform descriptive analysis. Understand the variables and their corresponding values. On the columns below, a value of zero does not make sense and thus indicates missing value:

• Glucose

• BloodPressure

• SkinThickness

• Insulin

• BMI

1. Visually explore these variables using histograms. Treat the missing values accordingly.
2. There are integer and float data type variables in this dataset. Create a count (frequency) plot describing the data types and the count of variables.

In [6]:

```
data.shape
```

Out[6]:

```
(768, 9)
```

In [7]:

```
data.dtypes
```

Out[7]:

```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

In [8]:

```
data.describe().T
```

Out[8]:

	count	mean	std	min	25%	50%	
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.0(
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.2!
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.0(
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.0(
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.2!
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.6(
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.6:
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.0(
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.0(

In [9]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [10]:

```
data_copy = data.copy(deep = True)
data_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = data_copy[['Gl
ucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)
```

In [11]:

```
data_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)
```

Out[11]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI
0	148.0	72.0	35.0	NaN	33.6
1	85.0	66.0	29.0	NaN	26.6
2	183.0	64.0	NaN	NaN	23.3
3	89.0	66.0	23.0	94.0	28.1
4	137.0	40.0	35.0	168.0	43.1
5	116.0	74.0	NaN	NaN	25.6
6	78.0	50.0	32.0	88.0	31.0
7	115.0	NaN	NaN	NaN	35.3
8	197.0	70.0	45.0	543.0	30.5
9	125.0	96.0	NaN	NaN	NaN
10	110.0	92.0	NaN	NaN	37.6
11	168.0	74.0	NaN	NaN	38.0
12	139.0	80.0	NaN	NaN	27.1
13	189.0	60.0	23.0	846.0	30.1
14	166.0	72.0	19.0	175.0	25.8
15	100.0	NaN	NaN	NaN	30.0
16	118.0	84.0	47.0	230.0	45.8
17	107.0	74.0	NaN	NaN	29.6
18	103.0	30.0	38.0	83.0	43.3
19	115.0	70.0	30.0	96.0	34.6
20	126.0	88.0	41.0	235.0	39.3
21	99.0	84.0	NaN	NaN	35.4
22	196.0	90.0	NaN	NaN	39.8
23	119.0	80.0	35.0	NaN	29.0
24	143.0	94.0	33.0	146.0	36.6
25	125.0	70.0	26.0	115.0	31.1
26	147.0	76.0	NaN	NaN	39.4
27	97.0	66.0	15.0	140.0	23.2
28	145.0	82.0	19.0	110.0	22.2
29	117.0	92.0	NaN	NaN	34.1
...	...	...	...	...	...
738	99.0	60.0	17.0	160.0	36.6
739	102.0	74.0	NaN	NaN	39.5
740	120.0	80.0	37.0	150.0	42.3
741	102.0	44.0	20.0	94.0	30.8
742	109.0	58.0	18.0	116.0	28.5
743	140.0	94.0	NaN	NaN	32.7
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

744	153.0	88.0	37.0	140.0	40.6

	Glucose	BloodPressure	SkinThickness	Insulin	BMI
745	100.0	84.0	33.0	105.0	30.0
746	147.0	94.0	41.0	NaN	49.3
747	81.0	74.0	41.0	57.0	46.3
748	187.0	70.0	22.0	200.0	36.4
749	162.0	62.0	NaN	NaN	24.3
750	136.0	70.0	NaN	NaN	31.2
751	121.0	78.0	39.0	74.0	39.0
752	108.0	62.0	24.0	NaN	26.0
753	181.0	88.0	44.0	510.0	43.3
754	154.0	78.0	32.0	NaN	32.4
755	128.0	88.0	39.0	110.0	36.5
756	137.0	90.0	41.0	NaN	32.0
757	123.0	72.0	NaN	NaN	36.3
758	106.0	76.0	NaN	NaN	37.5
759	190.0	92.0	NaN	NaN	35.5
760	88.0	58.0	26.0	16.0	28.4
761	170.0	74.0	31.0	NaN	44.0
762	89.0	62.0	NaN	NaN	22.5
763	101.0	76.0	48.0	180.0	32.9
764	122.0	70.0	27.0	NaN	36.8
765	121.0	72.0	23.0	112.0	26.2
766	126.0	60.0	NaN	NaN	30.1
767	93.0	70.0	31.0	NaN	30.4

768 rows × 5 columns

In [12]:

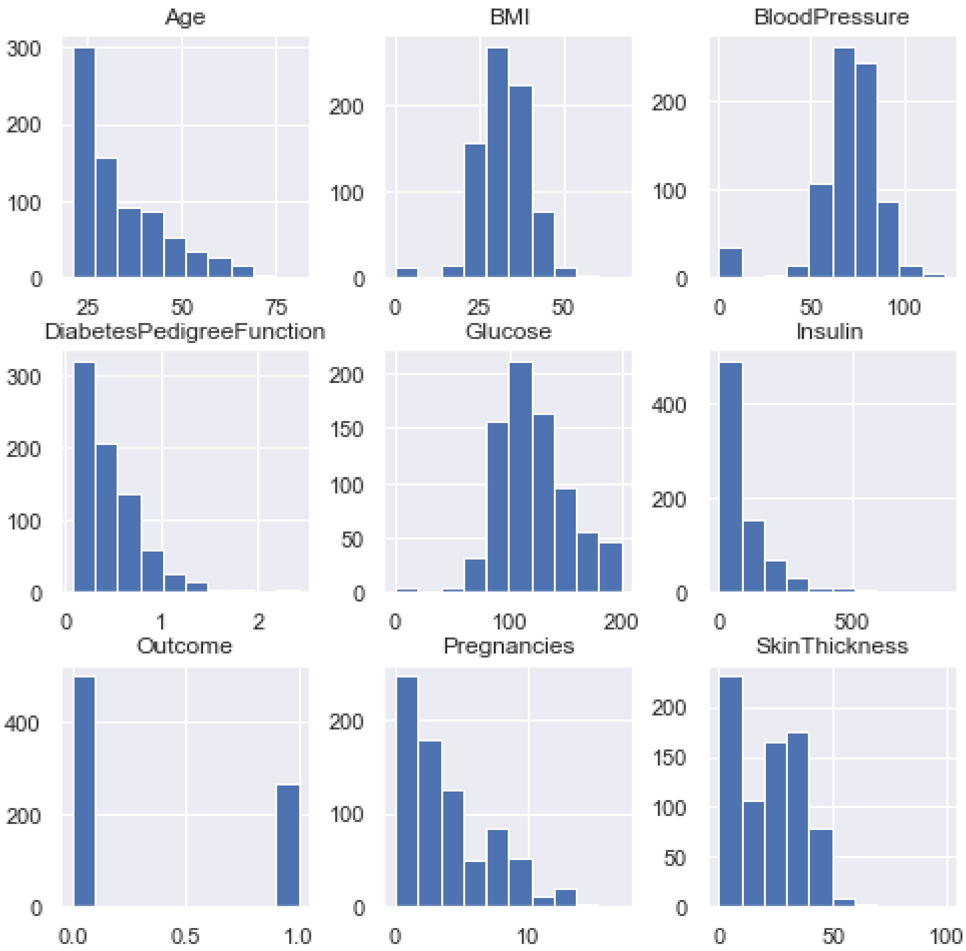
```
data_copy.isnull().sum()
```

Out[12]:

```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

In [13]:

```
# Creating Histgram
fig = plt.figure(figsize = (8,8))
ax = fig.gca()
data.hist(ax=ax)
plt.show()
```

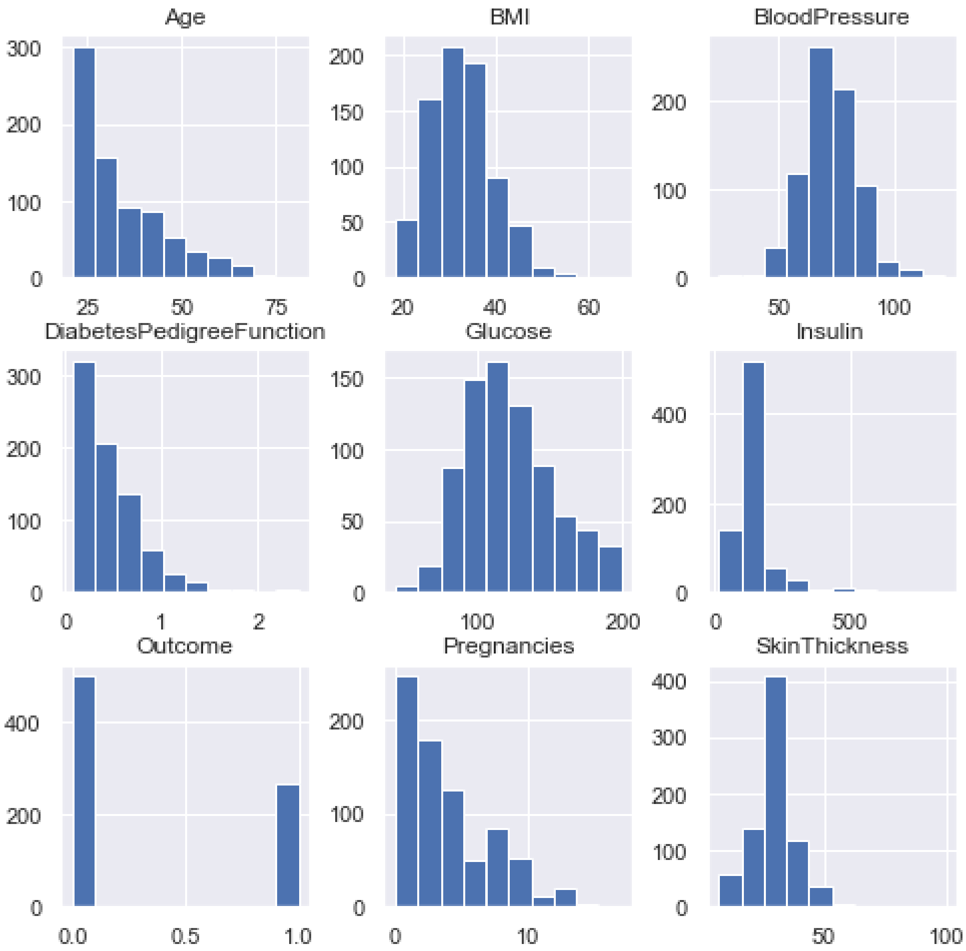


In [14]:

```
data_copy['Glucose'].fillna(data_copy['Glucose'].mean(), inplace = True)
data_copy['BloodPressure'].fillna(data_copy['BloodPressure'].mean(), inplace = True)
data_copy['SkinThickness'].fillna(data_copy['SkinThickness'].median(), inplace = True)
data_copy['Insulin'].fillna(data_copy['Insulin'].median(), inplace = True)
data_copy['BMI'].fillna(data_copy['BMI'].median(), inplace = True)
```

In [15]:

```
# Creating Histgram
fig = plt.figure(figsize = (8,8))
ax = fig.gca()
data_copy.hist(ax=ax)
plt.show()
```



In [16]:

```
data_copy.dtypes
```

Out[16]:

```
Pregnancies                   int64
Glucose                     float64
BloodPressure               float64
SkinThickness               float64
Insulin                     float64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

In [17]:

```
data.dtypes
```

Out[17]:

```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

In [18]:

```
 s = pd.Series(data.dtypes.values)
s.value_counts()

dtypes = pd.DataFrame(s,columns = ['Type'])
df =dtypes.groupby("Type").size().reset_index(name='counts')
```

In [19]:

```
df.set_index('Type', inplace=True)
```

In [20]:

```
df
```

Out[20]:

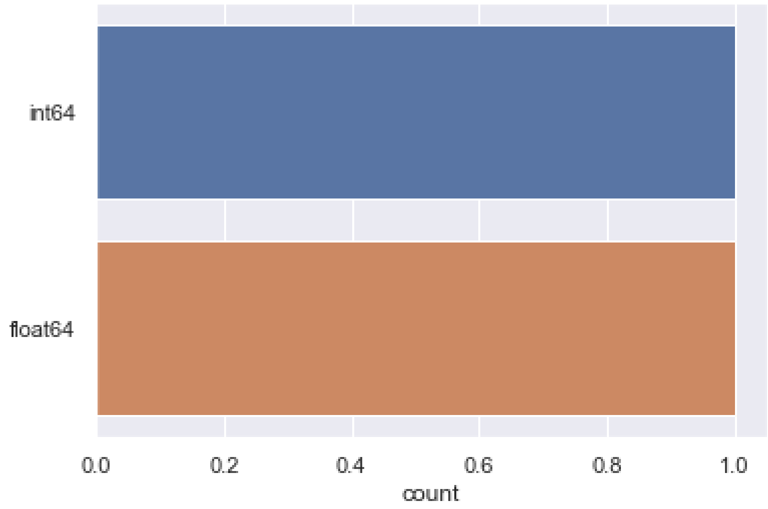
	counts
Type	
int64	7
float64	2

In [21]:

```
plt.setp(ax,yticks=[0,10])
sns.countplot(y=("int64", "float64"),
                 data=df)
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xb0bcb585f8>
```



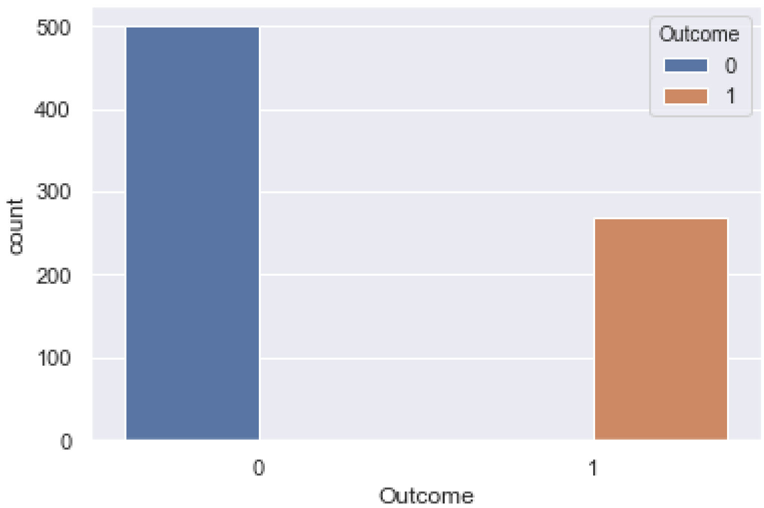
Project Task: Week 2

Data Exploration:

1. Check the balance of the data by plotting the count of outcomes by their value. Describe your findings and plan future course of action.
2. Create scatter charts between the pair of variables to understand the relationships. Describe your findings.
3. Perform correlation analysis. Visually explore it using a heat map.

In [22]:

```
sns.set(style="darkgrid")
ax = sns.countplot(x="Outcome", hue="Outcome" ,data=data_copy)
```



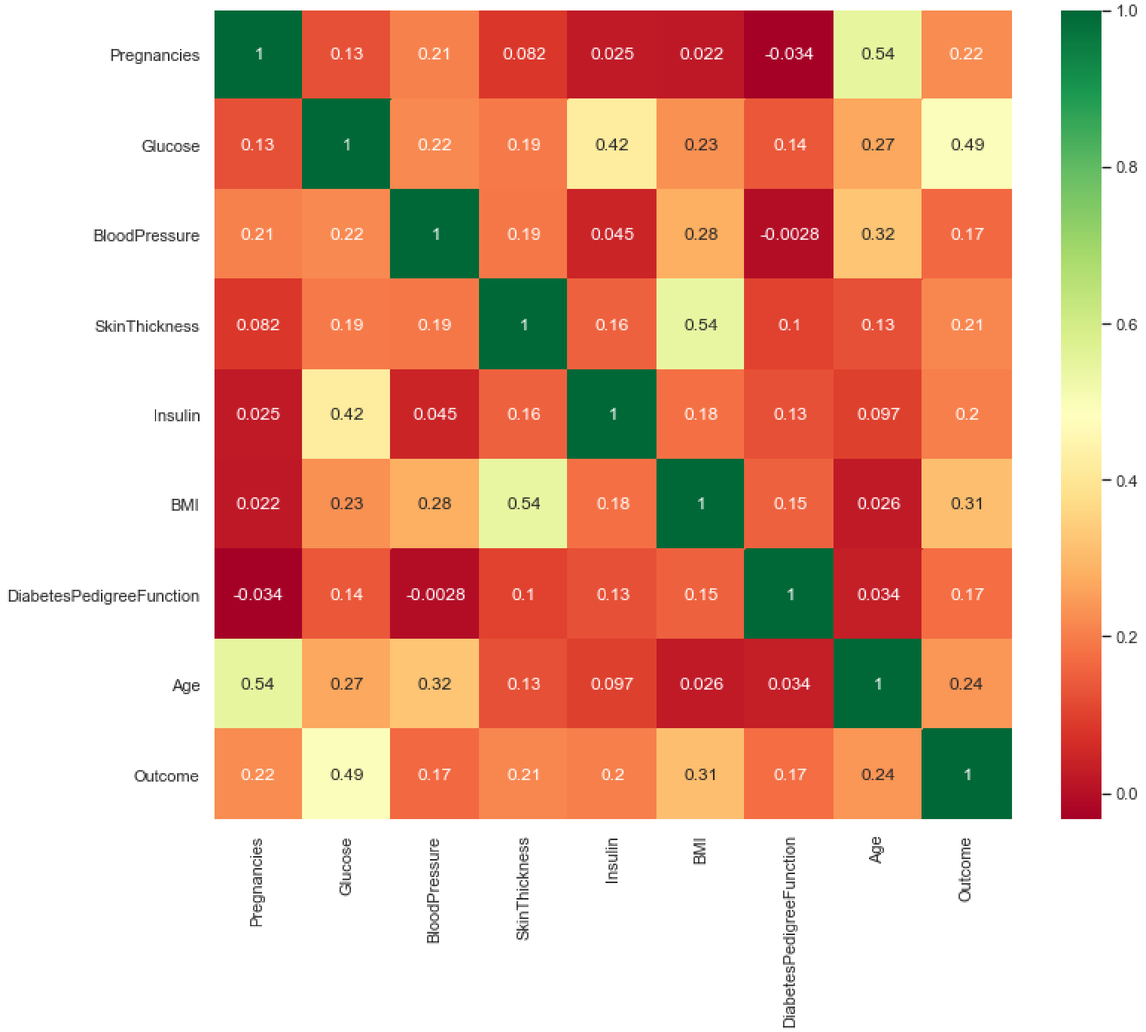
In [23]:

```
p=sns.pairplot(data_copy, hue = 'Outcome')
```

In [ ]:

In [24]:

```
plt.figure(figsize=(12,10))  # on this line I just set the size of figure to 12 by 10.
p=sns.heatmap(data_copy.corr(), annot=True,cmap ='RdYlGn')  # seaborn has very simple s
olution for heatmap
```



Project Task: Week 3

Data Modeling:

1. Devise strategies for model building. It is important to decide the right validation framework. Express your thought process.
2. Apply an appropriate classification algorithm to build a model. Compare various models with the results from KNN algorithm.

In [25]:

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X =  pd.DataFrame(sc_X.fit_transform(data_copy.drop(["Outcome"],axis = 1),),
        columns=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age'])
```

In [26]:

```
X.head()
```

Out[26]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedi
0	0.639947	0.865108	-0.033518	0.670643	-0.181541	0.166619	
1	-0.844885	-1.206162	-0.529859	-0.012301	-0.181541	-0.852200	
2	1.233880	2.015813	-0.695306	-0.012301	-0.181541	-1.332500	
3	-0.844885	-1.074652	-0.529859	-0.695245	-0.540642	-0.633881	
4	-1.141852	0.503458	-2.680669	0.670643	0.316566	1.549303	

In [27]:

```
y = data_copy.Outcome
```

In [28]:

```
#importing train_test_split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=1/3,random_state=42, str
atify=y)
```

In [40]:

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, cl
assification_report, confusion_matrix,fbeta_score

def print_results(headline, true_value, pred):
    print(headline)
    print("accuracy: {}".format(accuracy_score(true_value, pred)))
    print("precision: {}".format(precision_score(true_value, pred)))
    print("recall: {}".format(recall_score(true_value, pred)))
    print("f2: {}".format(fbeta_score(true_value, pred, beta=2)))
```

In [41]:

```
from sklearn.linear_model import LogisticRegression


# Create instance (i.e. object) of LogisticRegression
logmodel = LogisticRegression(C=10, penalty='l2',random_state=2)

# Fit the model using the training data
# X_train -> parameter supplies the data features
# y_train -> parameter supplies the target labels
logmodel.fit(X_train, y_train)

y_pred =logmodel.predict(X_test)


print_results("LogReg classification", y_test, y_pred)
```

```
LogReg classification
accuracy: 0.73046875
precision: 0.6351351351351351
recall: 0.5280898876404494
f2: 0.5465116279069767
```

In [43]:

```
from sklearn.neighbors import KNeighborsClassifier


test_scores = []
train_scores = []

for i in range(1,15):

    knn = KNeighborsClassifier(i)
    y_pred=knn.fit(X_train,y_train)

    train_scores.append(knn.score(X_train,y_train))
    test_scores.append(knn.score(X_test,y_test))
## score that comes from testing on the same datapoints that were used for training
max_train_score = max(train_scores)
train_scores_ind = [i for i, v in enumerate(train_scores) if v == max_train_score]
print('Max train score {} % and k = {}'.format(max_train_score*100,list(map(lambda x:
x+1, train_scores_ind))))
```

```
Max train score 100.0 % and k = [1]
```

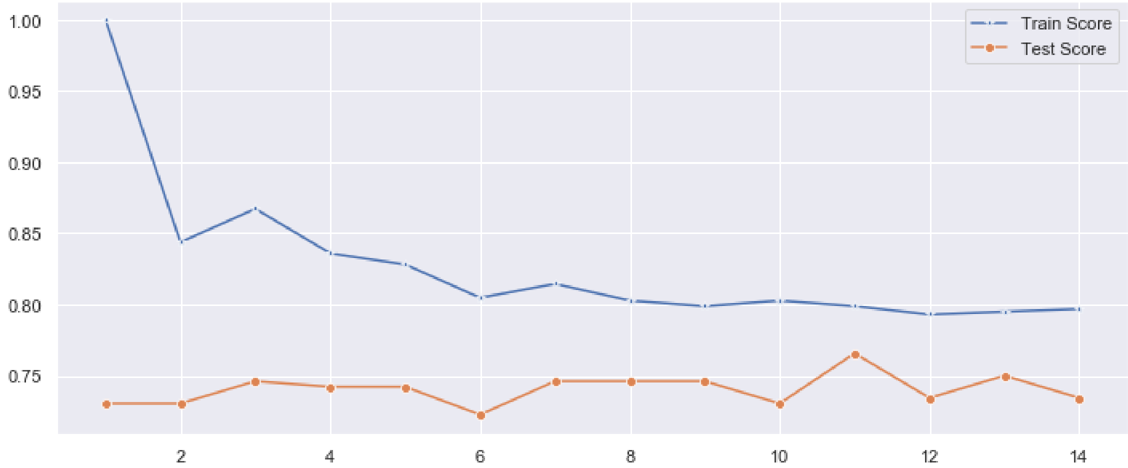
In [44]:

```
max_test_score = max(test_scores)
test_scores_ind = [i for i, v in enumerate(test_scores) if v == max_test_score]
print('Max test score {} % and k = {}'.format(max_test_score*100,list(map(lambda x:
x+1, test_scores_ind))))
```

```
Max test score 76.5625 % and k = [11]
```

In [45]:

```
plt.figure(figsize=(12,5))
p = sns.lineplot(range(1,15),train_scores,marker='*',label='Train Score')
p = sns.lineplot(range(1,15),test_scores,marker='o',label='Test Score')
```



In [46]:

```
#Setup a knn classifier with k neighbors
knn = KNeighborsClassifier(11)

knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

Out[46]:

```
0.765625
```

In [48]:

```
import confusion_matrix
from sklearn.metrics import confusion_matrix
#let us get the predictions using the classifier we had fit above
y_pred = knn.predict(X_test)
confusion_matrix(y_test,y_pred)
pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted'], margins=True)
print_results("KNN classification", y_test, y_pred)
```

```
KNN classification
accuracy: 0.765625
precision: 0.6835443037974683
recall: 0.6067415730337079
f2: 0.6206896551724138
```