In [73]:
```
# Name : Yvonne Lee
# Class: MSDS 680
# Assignment 2 Week 2
# Analysis:
# I first tested the KNN model with n = 1. The accuracy score came out to be 6
1%. I then plotted the accuracy scores and redid
# the KNN model with n = 5. The accuracy score after that became 63% which is
 a tad better than the first model. In an effort
# to continue to improve the model, I scaled the dataset and calculated accura
cy scores once more to plot. I then redid the
# KNN model again with n = 15 and that resulted in an accuracy score of 84%. T
here was a massive improvement after scaling
# the dataset. Each n for each step was the optimal k at the time of the mode
l.

# Import necessary libraries.
import pandas as pd
import numpy as np

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set()
```

In [74]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [75]:
```
# Upload dataset.
df = pd.read_csv("C:\\Users\\ylee_\\Desktop\\assign_wk2\\assign_wk2\\heart.disease.data.clean.csv")
df.head(10)
```

Out[75]:

	age	sex	cp	trestbps	chol	cigs	years	fbs	famhist	restecg	thalach	exang	thal	num
0	63	1	1	145	233	50.0	20.0	1	1	2	150	0	6	0
1	67	1	4	160	286	40.0	40.0	0	1	2	108	1	3	2
2	67	1	4	120	229	20.0	35.0	0	1	2	129	1	7	1
3	37	1	3	130	250	0.0	0.0	0	1	0	187	0	3	0
4	41	0	2	130	204	0.0	0.0	0	1	2	172	0	3	0
5	56	1	2	120	236	20.0	20.0	0	1	0	178	0	3	0
6	62	0	4	140	268	0.0	0.0	0	1	2	160	0	3	3
7	57	0	4	120	354	0.0	0.0	0	1	0	163	1	3	0
8	63	1	4	130	254	0.0	0.0	0	0	2	147	0	7	2
9	53	1	4	140	203	20.0	25.0	1	1	2	155	1	7	1

In [76]:
```
df.info()
```

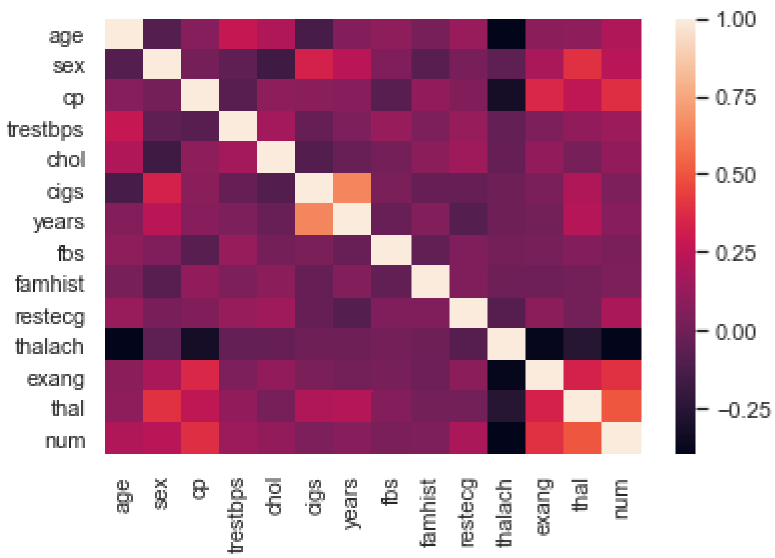
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 282 entries, 0 to 281
Data columns (total 14 columns):
age         282 non-null int64
sex         282 non-null int64
cp          282 non-null int64
trestbps    282 non-null int64
chol        282 non-null int64
cigs        282 non-null float64
years       282 non-null float64
fbs         282 non-null int64
famhist     282 non-null int64
restecg     282 non-null int64
thalach     282 non-null int64
exang       282 non-null int64
thal        282 non-null int64
num         282 non-null int64
dtypes: float64(2), int64(12)
memory usage: 31.0 KB
```

In [77]:
```
df.describe()
```

Out[77]:

	age	sex	cp	trestbps	chol	cigs	years	
count	282.000000	282.000000	282.000000	282.000000	282.000000	282.000000	282.000000	282.
mean	54.411348	0.677305	3.163121	131.195035	247.705674	16.836011	15.347364	0.
std	9.053083	0.468338	0.955405	16.739821	46.178771	18.876755	15.276814	0.
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	0.
25%	48.000000	0.000000	3.000000	120.000000	213.000000	0.000000	0.000000	0.
50%	55.000000	1.000000	3.000000	130.000000	244.000000	11.976385	15.000000	0.
75%	61.000000	1.000000	4.000000	140.000000	277.000000	30.000000	30.000000	0.
max	77.000000	1.000000	4.000000	170.000000	360.000000	75.000000	54.000000	1.

In [78]:
```
# Checking correlation.
_ = sns.heatmap(df.corr())
```



In [79]:
```
# Simplifying num values.
df["num"] = np.where(df["num"] >0,1,0)
```

In [80]:
```
df.head(10)
```

Out[80]:

	age	sex	cp	trestbps	chol	cigs	years	fbs	famhist	restecg	thalach	exang	thal	num
0	63	1	1	145	233	50.0	20.0	1	1	2	150	0	6	0
1	67	1	4	160	286	40.0	40.0	0	1	2	108	1	3	1
2	67	1	4	120	229	20.0	35.0	0	1	2	129	1	7	1
3	37	1	3	130	250	0.0	0.0	0	1	0	187	0	3	0
4	41	0	2	130	204	0.0	0.0	0	1	2	172	0	3	0
5	56	1	2	120	236	20.0	20.0	0	1	0	178	0	3	0
6	62	0	4	140	268	0.0	0.0	0	1	2	160	0	3	1
7	57	0	4	120	354	0.0	0.0	0	1	0	163	1	3	0
8	63	1	4	130	254	0.0	0.0	0	0	2	147	0	7	1
9	53	1	4	140	203	20.0	25.0	1	1	2	155	1	7	1

In [81]:
```
# Splitting the dataset to build a KNN model.
cols = df.columns
target_col = 'num'
feat_cols = [c for c in cols if c != target_col]
x = df[feat_cols].values
y = df[target_col].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

In [82]:
```
# KNN with n = 1.
model = KNeighborsClassifier(n_neighbors = 1, n_jobs = -1)
model.fit(x_train, y_train)
```

Out[82]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=1, p=2,
                     weights='uniform')
```

In [83]:
```
# Predict values.
preds = model.predict(x_test)
print('Actuals for test data set')
print(y_test)
print('Predictions for test data set')
print(preds)
```

```
Actuals for test data set
[1 0 1 1 0 1 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0 1
 0 0 1 1 1 1 0 0 1 0 1 0 1 1 0 1 1 0 0 1]
Predictions for test data set
[1 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0 0 0 1 0 0 1 0
 0 0 0 0 1 1 1 0 1 0 1 0 0 1 1 1 1 0 0 0]
```

In [84]:
```
differs = y_test - preds
print('Differences between the two sets')
print(differs)
```

```
Differences between the two sets
[ 0  0  0  1  0  1  0 -1  0  0  1  0  0  0  1  0  1  0  1  0 -1  0  0  0
 -1  0  0  0 -1  0  0  1 -1  1  1 -1  1  0  0  1  1  0  0 -1  0  0  0  0
  0  1  0 -1  0  0  0  0  1]
```

In [85]:
```
confusion_matrix(y_test, preds)
```

Out[85]:
```
array([[19,  8],
       [14, 16]], dtype=int64)
```

In [86]:
```
# We can see the summarized statistics for this model.
from sklearn.metrics import classification_report

print(classification_report(y_test,preds))
```

```
              precision    recall  f1-score   support

           0       0.58      0.70      0.63        27
           1       0.67      0.53      0.59        30

    accuracy                           0.61        57
   macro avg       0.62      0.62      0.61        57
weighted avg       0.62      0.61      0.61        57
```

In [88]:
```
# Confirm the accuracy score is 61%.
from sklearn.metrics import accuracy_score

print(accuracy_score(y_test,preds))
```

```
0.6140350877192983
```

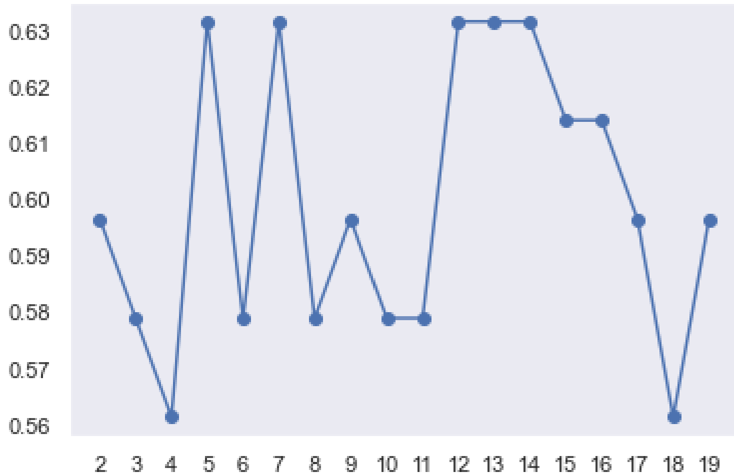
In [89]:
```python
# Getting accuracy scores to plot.
scores = []
print(f'Features: {feat_cols} \nTarget: {target_col}')

for k in range(2, 20):
    print(f'Evaluating {k} clusters')

    model = KNeighborsClassifier(n_neighbors=k, n_jobs=-1)
    model.fit(x_train, y_train)
    scores.append(model.score(x_test, y_test))
```

```
Features: ['age', 'sex', 'cp', 'trestbps', 'chol', 'cigs', 'years', 'fbs', 'f
amhist', 'restecg', 'thalach', 'exang', 'thal']
Target: num
Evaluating 2 clusters
Evaluating 3 clusters
Evaluating 4 clusters
Evaluating 5 clusters
Evaluating 6 clusters
Evaluating 7 clusters
Evaluating 8 clusters
Evaluating 9 clusters
Evaluating 10 clusters
Evaluating 11 clusters
Evaluating 12 clusters
Evaluating 13 clusters
Evaluating 14 clusters
Evaluating 15 clusters
Evaluating 16 clusters
Evaluating 17 clusters
Evaluating 18 clusters
Evaluating 19 clusters
```

In [90]:
```python
scores
```

Out[90]:
```
[0.5964912280701754,
 0.5789473684210527,
 0.5614035087719298,
 0.631578947368421,
 0.5789473684210527,
 0.631578947368421,
 0.5789473684210527,
 0.5964912280701754,
 0.5789473684210527,
 0.5789473684210527,
 0.631578947368421,
 0.631578947368421,
 0.631578947368421,
 0.6140350877192983,
 0.6140350877192983,
 0.5964912280701754,
 0.5614035087719298,
 0.5964912280701754]
```

In [91]:
```
# Plotting accuracy scores.
plt.plot(range(2, 20), scores)
plt.scatter(range(2, 20), scores)
plt.grid()
_ =plt.xticks(range(2, 20))
```



In [92]:
```
# KNN
model = KNeighborsClassifier(n_neighbors=5, n_jobs=-1)
model.fit(x_train, y_train)

preds = model.predict(x_test)

print('Actuals for test data set')
print(y_test)
print('Predictions for test data set')
print(preds)
```

```
Actuals for test data set
[1 0 1 1 0 1 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0 1
 0 0 1 1 1 1 0 0 1 0 1 0 1 1 0 1 1 0 0 1]
Predictions for test data set
[0 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0 1 0 0 0 0
 0 1 0 1 1 1 1 0 1 0 1 0 0 1 0 1 1 0 0 0]
```

In [93]:
```
# CHecking stats and accuracy score is 63%.
differs = y_test - preds

print(f'Differences between the two sets:\n{differs}')

print(f'accuracy score: {accuracy_score(y_test,preds)}')
```

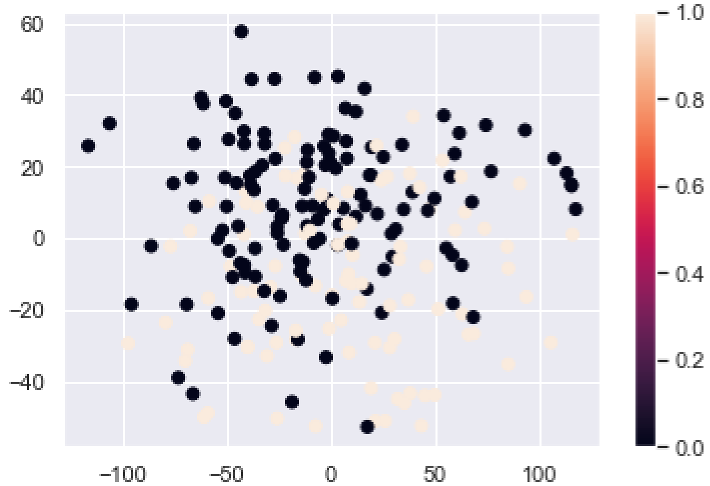
```
Differences between the two sets:
[ 1  0  0  0  0  0  0 -1  1 -1  1  0  0  0  1  0  0  1  0  0 -1  0  1  0
  0  1  0  0  0  0 -1  1 -1  1  1  0  1  0 -1  1  0  0  0 -1  0  0  0  0
  0  1  0  0  0  0  0  0  1]
accuracy score: 0.631578947368421
```

In [94]:
```
confusion_matrix(y_test, preds)
```

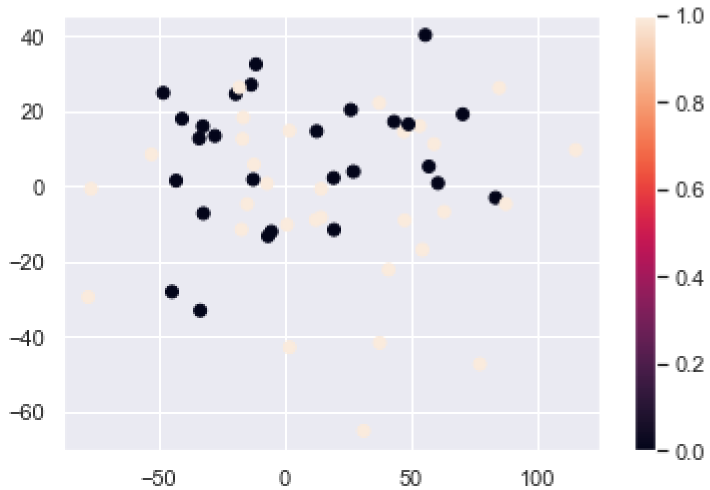
Out[94]:
```
array([[20,  7],
       [14, 16]], dtype=int64)
```

In [95]:
```
pc = PCA()
tr_pca = pc.fit_transform(x_train)
te_pca = pc.transform(x_test)
```

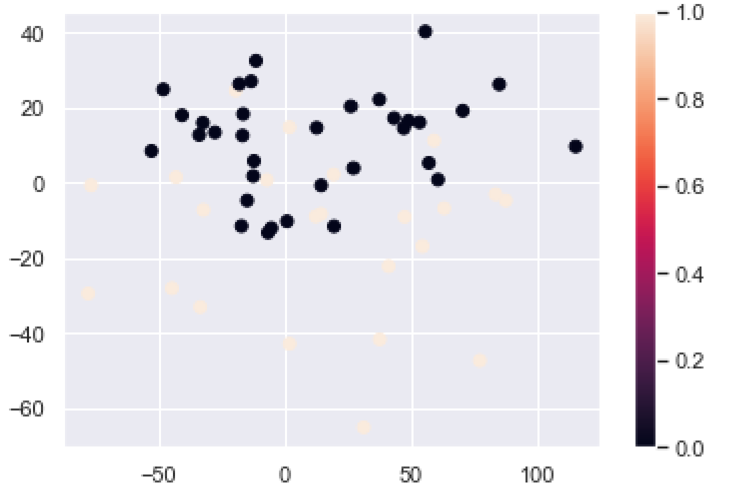
In [96]:
```
_ = plt.scatter(tr_pca[:, 0], tr_pca[:, 1], c=y_train)
_ = plt.colorbar()
```



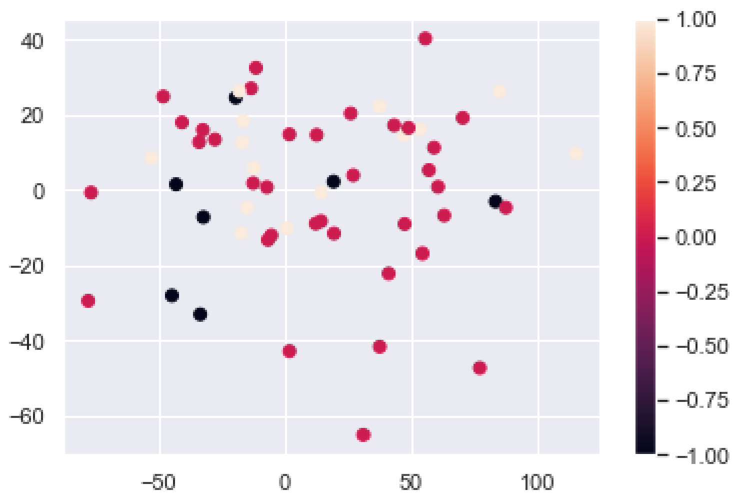
In [97]:
```
_ = plt.scatter(te_pca[:, 0], te_pca[:, 1], c=y_test)
_ = plt.colorbar()
```



In [98]:
```
_ = plt.scatter(te_pca[:, 0], te_pca[:, 1], c=preds)
_ = plt.colorbar()
```



In [99]:
```
_ = plt.scatter(te_pca[:, 0], te_pca[:, 1], c=differs)
_ = plt.colorbar()
```



In [100]:
```
df.describe()
```

Out[100]:

	age	sex	cp	trestbps	chol	cigs	years	
count	282.000000	282.000000	282.000000	282.000000	282.000000	282.000000	282.000000	282.
mean	54.411348	0.677305	3.163121	131.195035	247.705674	16.836011	15.347364	0.
std	9.053083	0.468338	0.955405	16.739821	46.178771	18.876755	15.276814	0.
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	0.
25%	48.000000	0.000000	3.000000	120.000000	213.000000	0.000000	0.000000	0.
50%	55.000000	1.000000	3.000000	130.000000	244.000000	11.976385	15.000000	0.
75%	61.000000	1.000000	4.000000	140.000000	277.000000	30.000000	30.000000	0.
max	77.000000	1.000000	4.000000	170.000000	360.000000	75.000000	54.000000	1.

In [101]:
```
df.head(10)
```

Out[101]:

	age	sex	cp	trestbps	chol	cigs	years	fbs	famhist	restecg	thalach	exang	thal	num
0	63	1	1	145	233	50.0	20.0	1	1	2	150	0	6	0
1	67	1	4	160	286	40.0	40.0	0	1	2	108	1	3	1
2	67	1	4	120	229	20.0	35.0	0	1	2	129	1	7	1
3	37	1	3	130	250	0.0	0.0	0	1	0	187	0	3	0
4	41	0	2	130	204	0.0	0.0	0	1	2	172	0	3	0
5	56	1	2	120	236	20.0	20.0	0	1	0	178	0	3	0
6	62	0	4	140	268	0.0	0.0	0	1	2	160	0	3	1
7	57	0	4	120	354	0.0	0.0	0	1	0	163	1	3	0
8	63	1	4	130	254	0.0	0.0	0	0	2	147	0	7	1
9	53	1	4	140	203	20.0	25.0	1	1	2	155	1	7	1

In [102]:
```
# Scaling the dataset.
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x_train)

x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

In [104]:
```
new_df_tr = pd.DataFrame(x_train, columns=feat_cols)
new_df_tr['num'] = y_train
new_df_tr.head(10)
```

Out[104]:

	age	sex	cp	trestbps	chol	cigs	years	fbs	famhi
0	-2.726464	0.658119	-1.183867	-0.058320	-0.891198	-0.808009	-0.649990	-0.414578	-1.3205
1	-1.849944	-1.519481	-0.147408	-0.671498	-0.654050	-0.913716	-1.051226	-0.414578	0.7572
2	1.437006	-1.519481	0.889052	-1.529947	-0.481579	0.143348	2.292409	-0.414578	-1.3205
3	0.231791	0.658119	-2.220326	-0.671498	-1.128346	1.200413	1.489937	-0.414578	0.7572
4	-0.535164	0.658119	-0.147408	-0.794134	-2.076939	0.407615	1.623682	-0.414578	0.7572
5	2.203960	-1.519481	-1.183867	-0.671498	0.510132	-0.913716	-1.051226	-0.414578	0.7572
6	-0.096904	0.658119	0.889052	0.677494	-0.416902	1.200413	0.954955	-0.414578	0.7572
7	0.889181	-1.519481	0.889052	0.432222	1.049105	0.143348	0.620592	2.412091	0.7572
8	0.560486	0.658119	-1.183867	0.554858	-0.524696	0.143348	1.623682	-0.414578	0.7572
9	-0.973424	-1.519481	-1.183867	-1.162040	-1.839791	-0.913716	-1.051226	-0.414578	-1.3205

In [105]:
```
new_df_te = pd.DataFrame(x_test, columns=feat_cols)
new_df_te['num'] = y_test
new_df_te.head(10)
```

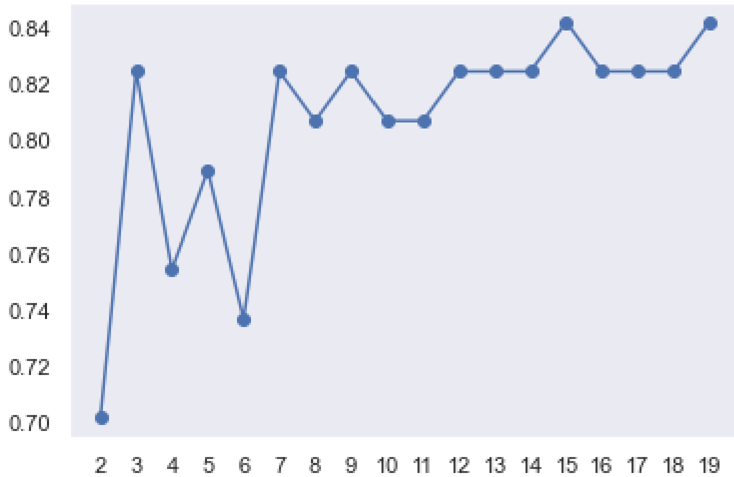
Out[105]:

	age	sex	cp	trestbps	chol	cigs	years	fbs	famhi
0	-1.192554	0.658119	0.889052	0.064316	0.035836	0.936147	0.286228	2.412091	0.7572
1	0.012661	-1.519481	-0.147408	1.781214	-0.955875	-0.913716	-1.051226	-0.414578	0.7572
2	0.889181	-1.519481	0.889052	1.781214	-1.753555	-0.913716	-1.051226	-0.414578	0.7572
3	-0.316034	0.658119	0.889052	0.554858	1.135341	-0.913716	-1.051226	-0.414578	-1.3205
4	0.012661	-1.519481	-0.147408	0.248269	1.264695	2.257477	0.954955	2.412091	-1.3205
5	0.779616	-1.519481	0.889052	0.861447	1.329371	0.143348	0.620592	-0.414578	0.7572
6	0.012661	0.658119	-1.183867	-1.407312	1.372489	0.671881	1.222446	-0.414578	-1.3205
7	-1.302119	0.658119	0.889052	0.554858	-0.416902	0.143348	0.620592	-0.414578	0.7572
8	0.560486	0.658119	-2.220326	2.394392	0.919752	-0.913716	-1.051226	-0.414578	-1.3205
9	0.560486	0.658119	-0.147408	1.168036	-0.718727	1.728945	0.954955	2.412091	-1.3205

In [107]:
```
# New accuracy scores after dataset is scaled.
scores_norm = []
print(f'Features: {feat_cols} \nTarget: {target_col}')

for k in range(2, 20):
    print(f'Evaluating {k} clusters')

    model_norm = KNeighborsClassifier(n_neighbors=k, n_jobs=-1)
    model_norm.fit(x_train, y_train)
    scores_norm.append(model_norm.score(x_test, y_test))
```

```
Features: ['age', 'sex', 'cp', 'trestbps', 'chol', 'cigs', 'years', 'fbs', 'f
amhist', 'restecg', 'thalach', 'exang', 'thal']
Target: num
Evaluating 2 clusters
Evaluating 3 clusters
Evaluating 4 clusters
Evaluating 5 clusters
Evaluating 6 clusters
Evaluating 7 clusters
Evaluating 8 clusters
Evaluating 9 clusters
Evaluating 10 clusters
Evaluating 11 clusters
Evaluating 12 clusters
Evaluating 13 clusters
Evaluating 14 clusters
Evaluating 15 clusters
Evaluating 16 clusters
Evaluating 17 clusters
Evaluating 18 clusters
Evaluating 19 clusters
```

In [108]:
```
# New accuracy score plot.
plt.plot(range(2, 20), scores_norm)
plt.scatter(range(2, 20), scores_norm)
plt.grid()
_ =plt.xticks(range(2, 20))
```



In [110]:
```
# KNN with the new n value based on the plot.
model_norm = KNeighborsClassifier(n_neighbors=15, n_jobs=-1)
model_norm.fit(x_train, y_train)

preds_norm = model_norm.predict(x_test)

print('Actuals for test data set')
print(y_test)
print('Predictions for test data set')
print(preds_norm)
```

```
Actuals for test data set
[1 0 1 1 0 1 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0 1
 0 0 1 1 1 1 0 0 1 0 1 0 1 1 0 1 1 0 0 1]
Predictions for test data set
[1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 1 0 0
 0 0 0 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 0 0]
```

In [111]:
```
# Checking the new accuracy score.
differs_norm = y_test - preds_norm

print(f'Differences between the two sets:\n{differs_norm}')

print(f'accuracy score: {accuracy_score(y_test,preds_norm)}')
```

```
Differences between the two sets:
[ 0  0  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0  1  0  0  0  0  1  0
  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  1  0  0  0  0  0  0  0  0
  0  0  0 -1  0  0  0  0  1]
accuracy score: 0.8421052631578947
```

In [ ]: