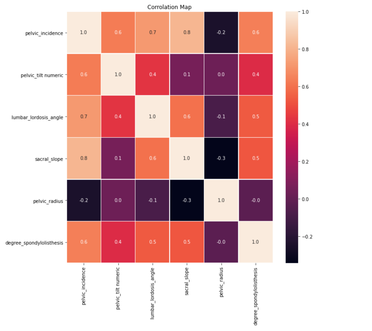
In [2]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns

#READ DATA
data = pd.read_csv("column_2C_weka.csv")
data.head()
```

Out[2]:

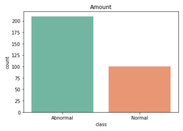
	pelvic_incidence	pelvic_tilt numeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis	class
0	63.027818	22.552586	39.609117	40.475232	98.672917	-0.254400	Abnormal
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.564259	Abnormal
2	68.832021	22.218482	50.092194	46.613539	105.985135	-3.530317	Abnormal
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.211523	Abnormal
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.918501	Abnormal

In [3]:
```
#VISUALIZATION

f,ax = plt.subplots(figsize=(10, 10))
sns.heatmap(data.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
plt.title("Correlation Map")
plt.show()
data["class"].value_counts()
```

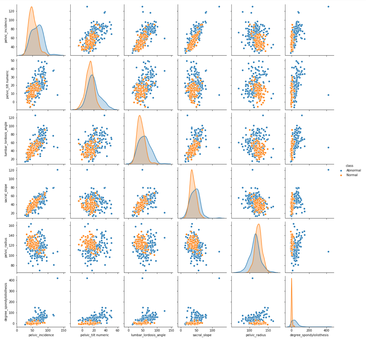


Out[3]:
```
Abnormal    210
Normal      100
Name: class, dtype: int64
```

In [4]:
```
sns.countplot(data["class"], palette="Set2")
plt.title("Amount")
plt.show()
```



In [5]:
```
sns.pairplot(data=data, hue="class")
plt.show()
```



In [6]:
```
#DATA PREPROCESSING

data["class"] = [0 if i == "Abnormal" else 1 for i in data["class"]]
y = data["class"].values
```

In [9]:
```
x_data = data.drop(["class"], axis=1)
```

In [10]:
```
x = (x_data - np.min(x_data))/(np.max(x_data) - np.min(x_data)).values
x
```

Out[10]:

	pelvic_incidence	pelvic_tilt numeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis
0	0.355688	0.519900	0.229180	0.250857	0.307461	0.025148
1	0.124501	0.296783	0.098578	0.144629	0.476649	0.036365
2	0.411666	0.513932	0.322995	0.307661	0.386097	0.017523
3	0.416151	0.557414	0.271260	0.289436	0.341826	0.051838
4	0.227272	0.289479	0.128129	0.247022	0.409579	0.044173
...	...	...	...	...	...	...
305	0.209822	0.360293	0.196881	0.193591	0.509380	0.015858
306	0.268009	0.487194	0.136211	0.183674	0.476223	0.024761
307	0.340438	0.522443	0.287897	0.234907	0.597796	0.019437
308	0.184257	0.272352	0.246846	0.214623	0.521175	0.026240
309	0.074202	0.207709	0.202620	0.142517	0.579240	0.025277

310 rows × 6 columns

In [11]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=11)
```

In [12]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5) #k value
knn.fit(x_train, y_train)
print(" {} nn score: {} ".format(5,knn.score(x_test,y_test)))
```

```
 5 nn score: 0.8870967741935484
```

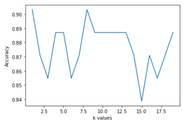
In [13]:
```
score_list = []
for i in range(1,20):
    knn2 = KNeighborsClassifier(n_neighbors = i)
    knn2.fit(x_train, y_train)
    score_list.append(knn2.score(x_test, y_test))
score_list
```

Out[13]:
```
[0.9032258064516129,
 0.8709677419354839,
 0.8548387096774194,
 0.8870967741935484,
 0.8870967741935484,
 0.8548387096774194,
 0.8709677419354839,
 0.9032258064516129,
 0.8870967741935484,
 0.8870967741935484,
 0.8870967741935484,
 0.8870967741935484,
 0.8709677419354839,
 0.8387096774193549,
 0.8709677419354839,
 0.8548387096774194,
 0.8709677419354839,
 0.8870967741935484]
```

In [14]:
```
current_max = score_list[0]
for i in range(len(score_list)):
    if current_max < score_list[i]:
        current_max = score_list[i]
current_max
```

Out[14]:
```
0.9032258064516129
```

In [15]:
```
plt.plot(range(1,20),score_list)
plt.xlabel("k values")
plt.ylabel("Accuracy")
plt.show()
```



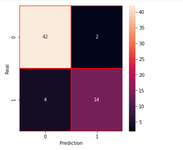
In [16]:
```
knn3 = KNeighborsClassifier(n_neighbors = 8) #k value
knn3.fit(x_train, y_train)
print(" {} nn score: {} ".format(8,knn3.score(x_test,y_test)))
```

```
 8 nn score: 0.9032258064516129
```

In [17]:
```
y_pred_knn = knn3.predict(x_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred_knn)
print("KNN result confusion matrix : \n", cm)
```

```
KNN result confusion matrix :
 [[42  2]
 [ 4 14]]
```

In [18]:
```
#confusion metrics visualization
f, ax = plt.subplots(figsize=(5,5))
sns.heatmap(cm, annot=True, linewidths=0.5, linecolor="red", fmt=".0f", ax=ax)
plt.xlabel("Prediction")
plt.ylabel("Real")
plt.show()
```



In [ ]: