In [28]:

```
import pandas as pd
import matplotlib.pyplot as plt
```

In [29]:

```
dataFrame = pd.read_table("[redacted]Data set/column_2C.dat", sep="\s+
```

In [30]:

```
dataFrame.columns =['Pelvic incidence','Pelvic tilt','Lumbar lordosis angle','Sacral slope'
                    'Grade of Spondylolisthesis','target']
```

In [31]:

```
dataFrame.head(n=7)
```

Out[31]:

	Pelvic incidence	Pelvic tilt	Lumbar lordosis angle	Sacral slope	Pelvic radius	Grade of Spondylolisthesis	target
0	63.03	22.55	39.61	40.48	98.67	-0.25	AB
1	39.06	10.06	25.02	29.00	114.41	4.56	AB
2	68.83	22.22	50.09	46.61	105.99	-3.53	AB
3	69.30	24.65	44.31	44.64	101.87	11.21	AB
4	49.71	9.65	28.32	40.06	108.17	7.92	AB
5	40.25	13.92	25.12	26.33	130.33	2.23	AB
6	53.43	15.86	37.17	37.57	120.57	5.99	AB

In [32]:

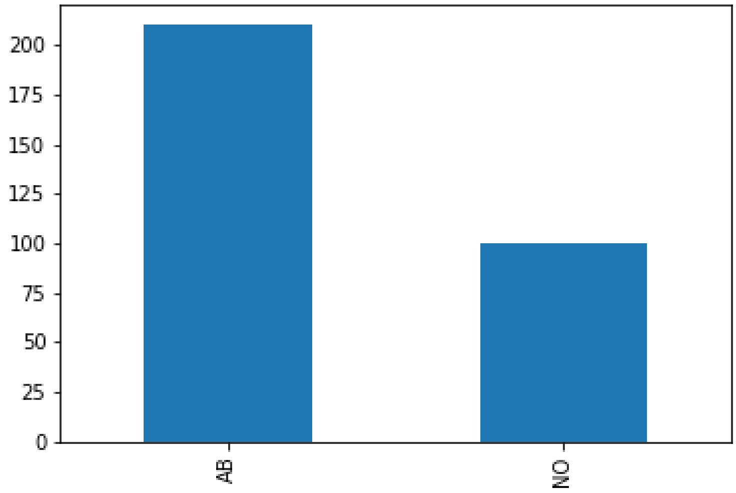
```
dataFrame.shape
```

Out[32]:

```
(310, 7)
```

In [33]:

```
dataFrame.target.value_counts().plot(kind='bar')
plt.show()
```



In [34]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dataFrame.target = le.fit_transform(dataFrame.target)
```

In [35]:

```
dataFrame.target.value_counts()
```

Out[35]:

```
0    210
1    100
Name: target, dtype: int64
```

In [36]:

```
import numpy as np
```

In [37]:

```
corr = dataFrame.corr()
```

In [38]:

```
corr.style.background_gradient(cmap='coolwarm')
```

Out[38]:

	Pelvic incidence	Pelvic tilt	Lumbar lordosis angle	Sacral slope	Pelvic radius	Grade of Spondylolisthesis	t
Pelvic incidence	1.000000	0.629186	0.717286	0.814959	-0.247484	0.638733	-0.35
Pelvic tilt	0.629186	1.000000	0.432760	0.062327	0.032660	0.397840	-0.32
Lumbar lordosis angle	0.717286	0.432760	1.000000	0.598389	-0.080368	0.533665	-0.31
Sacral slope	0.814959	0.062327	0.598389	1.000000	-0.342147	0.523571	-0.21
Pelvic radius	-0.247484	0.032660	-0.080368	-0.342147	1.000000	-0.026073	0.30
Grade of Spondylolisthesis	0.638733	0.397840	0.533665	0.523571	-0.026073	1.000000	-0.44
target	-0.353323	-0.326048	-0.312492	-0.210611	0.309875	-0.443682	1.00

In [39]:

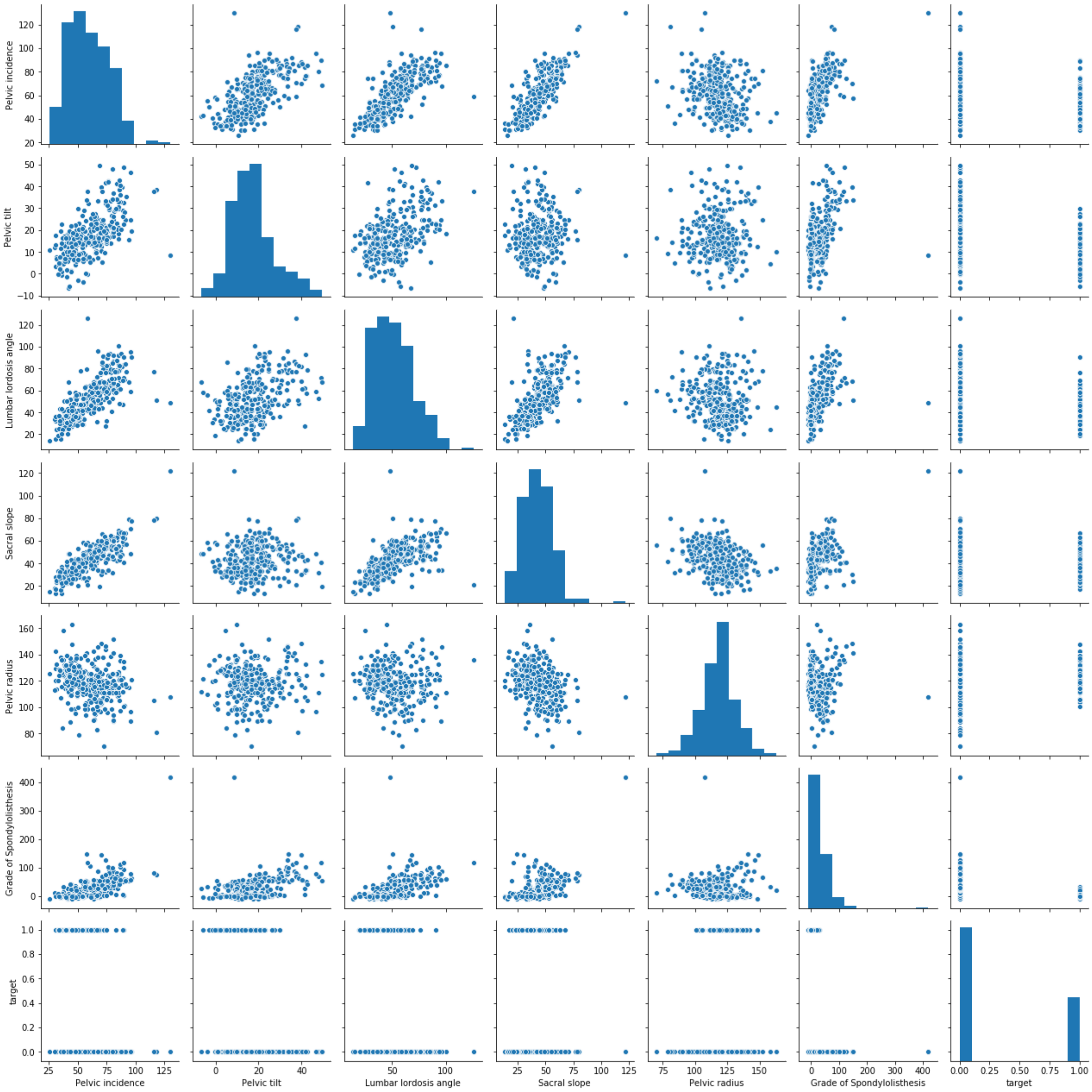
```
import seaborn as sns
```

In [40]:

```
sns.pairplot(dataFrame)
```

Out[40]:

```
<seaborn.axisgrid.PairGrid at 0x275b019da08>
```



In [41]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(dataFrame.drop('target',axis=1), dataFr
```

In [42]:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, accuracy_score
gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train).predict(X_test)
print('Accuracy of model is ' , accuracy_score(y_pred,y_test)*100)
```

```
Accuracy of model is  79.03225806451613
```

In [62]:

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
logistic_regression = LogisticRegression()
logistic_regression.fit(X_train, y_train)
```

Out[62]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [63]:

```
y_pred = logistic_regression.predict(X_test)
accuracy = metrics.accuracy_score(y_test, y_pred)
accuracy_percentage = 100 * accuracy
accuracy_percentage
```

Out[63]:

```
85.48387096774194
```

In [46]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [47]:

```
# for K value = 5
knnClassifer = KNeighborsClassifier(n_neighbors=5)
knnClassifer.fit(X_train, y_train)
knnPrediction = knnClassifer.predict(X_test)
```

In [48]:

```
print('\n\nClassification report')
print(classification_report(y_test, knnPrediction))
print('Accuracy when K=5 is' , accuracy_score(knnPrediction,y_test)*100)
```

```


Classification report
              precision    recall  f1-score   support

           0       0.89      0.89      0.89        44
           1       0.72      0.72      0.72        18

    accuracy                           0.84        62
   macro avg       0.80      0.80      0.80        62
weighted avg       0.84      0.84      0.84        62

Accuracy when K=5 is 83.87096774193549
```

In [49]:

```
# for K value = 7
knnClassifer = KNeighborsClassifier(n_neighbors=7)
knnClassifer.fit(X_train, y_train)
knnPrediction = knnClassifer.predict(X_test)
print('\n\nClassification report')
print(classification_report(y_test, knnPrediction))
print('Accuracy when K=7 is' , accuracy_score(knnPrediction,y_test)*100)
```

```


Classification report
              precision    recall  f1-score   support

           0       0.87      0.89      0.88        44
           1       0.71      0.67      0.69        18

    accuracy                           0.82        62
   macro avg       0.79      0.78      0.78        62
weighted avg       0.82      0.82      0.82        62

Accuracy when K=7 is 82.25806451612904
```

In [58]:

```
# for K value = 10
knnClassifer = KNeighborsClassifier(n_neighbors=10)
knnClassifer.fit(X_train, y_train)
knnPrediction = knnClassifer.predict(X_test)
print('\n\nClassification report')
print(classification_report(y_test, knnPrediction))
print('Accuracy when K=10 is' , accuracy_score(knnPrediction,y_test)*100)
```

```


Classification report
              precision    recall  f1-score   support

           0       0.85      0.89      0.87        44
           1       0.69      0.61      0.65        18

    accuracy                           0.81        62
   macro avg       0.77      0.75      0.76        62
weighted avg       0.80      0.81      0.80        62

Accuracy when K=10 is 80.64516129032258
```

In [68]:

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, accuracy_score
rf = RandomForestClassifier()
rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
print('Accuracy of model is ' , accuracy_score(y_pred,y_test)*100)
```

```
Accuracy of model is  80.64516129032258
```

In [69]:

```
print(classification_report(y_pred,y_test))
```

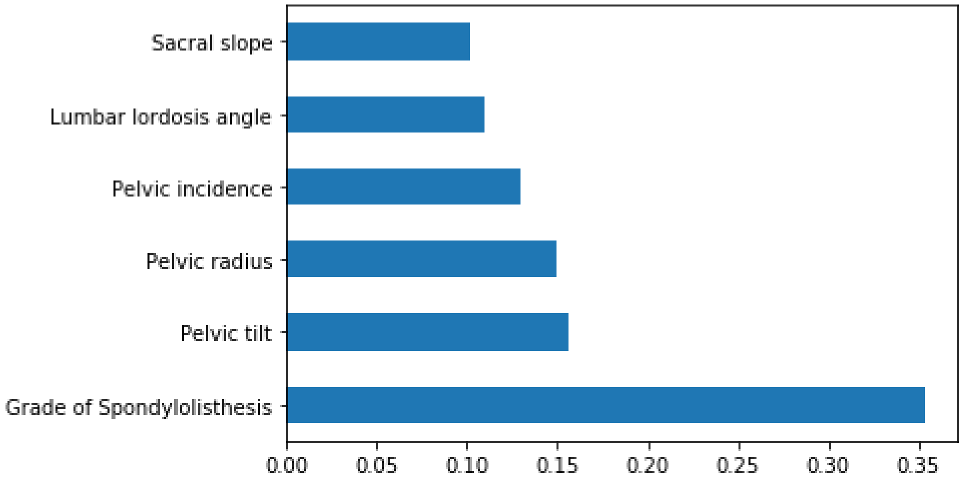
```
              precision    recall  f1-score   support

           0       0.89      0.85      0.87        46
           1       0.61      0.69      0.65        16

    accuracy                           0.81        62
   macro avg       0.75      0.77      0.76        62
weighted avg       0.82      0.81      0.81        62
```

In [70]:

```
featureImportances = pd.Series(rf.feature_importances_, index=dataFrame.drop('target',axis=
featureImportances.nlargest(6).plot(kind='barh')
```

Out[70]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x275b33799c8>
```



In [ ]:

```

```