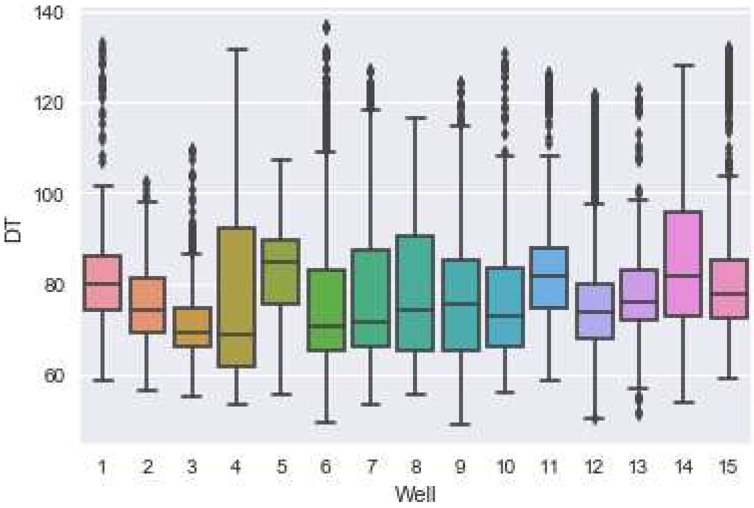
```
In [22]: %matplotlib inline
         import matplotlib
         import matplotlib.pyplot as plt
         import numpy as np
         import seaborn as sns; sns.set()
         import pandas as pd
```

```
In [23]: data1 = pd.read_csv("Wells.csv")
```

```
In [24]: data1.shape

Out[24]: (55945, 22)
```

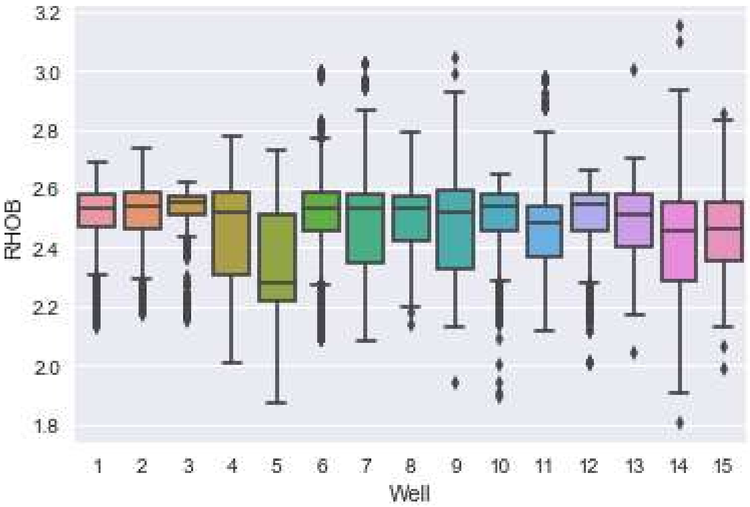
```
In [25]: data2=data1[['Well','GR']]
```

```
In [26]: sns.boxplot(x=data1['Well'],y=data1['DT'])

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x22a43c3e550>
```



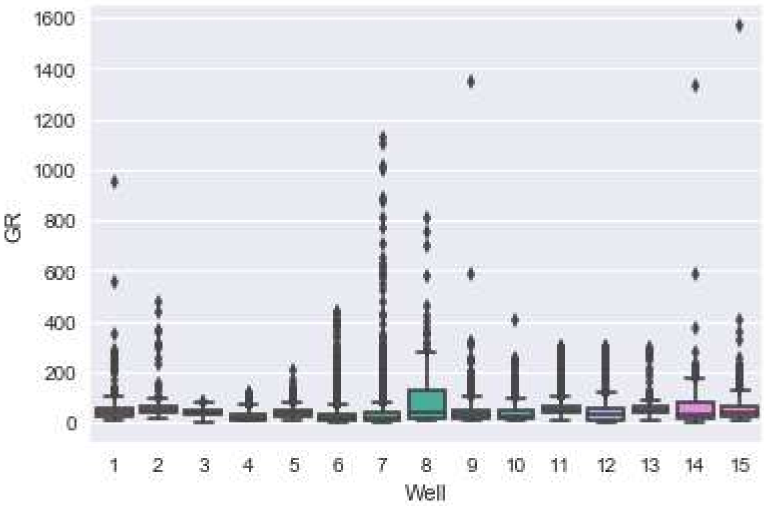
In [27]:
```
sns.boxplot(x=data1['Well'],y=data1['RHOB'])
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22a448be128>
```



In [28]:
```
sns.boxplot(x=data2['Well'],y=data2['GR'])
```

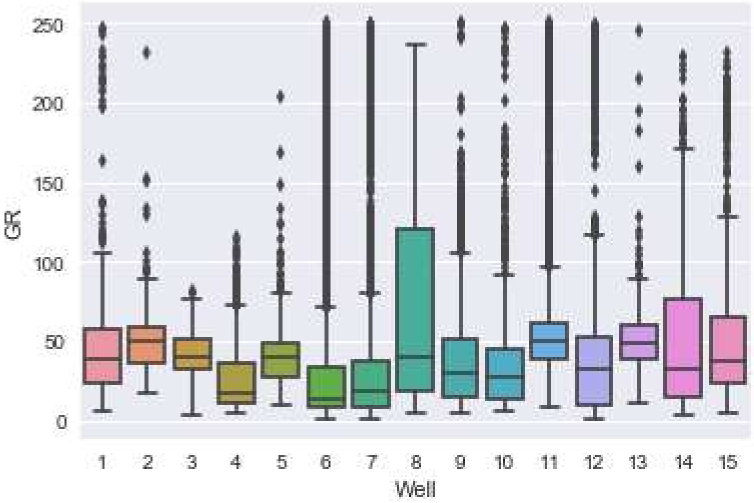
Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x22a44a2d6a0>
```



In [29]:
```
data3=data2[(data2['GR']<250)]
```

```
In [30]: sns.boxplot(x=data3['Well'],y=data3['GR'])
```

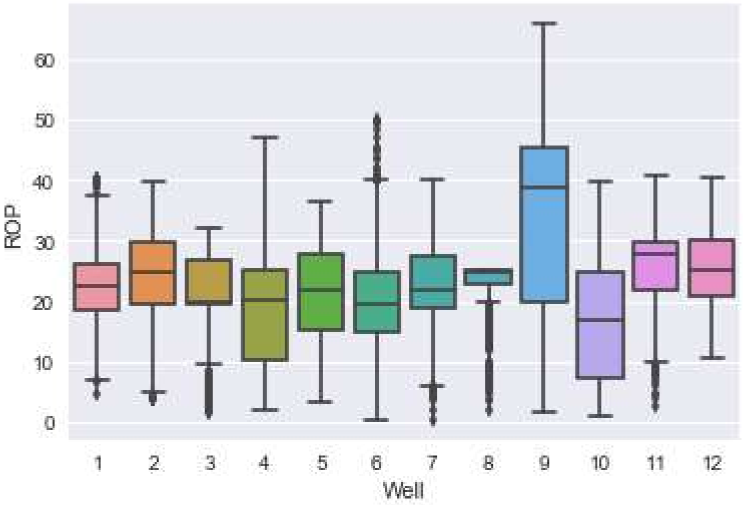
```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x22a44b9b898>
```



```
In [31]: d=data1[data1['Well']<13]
```

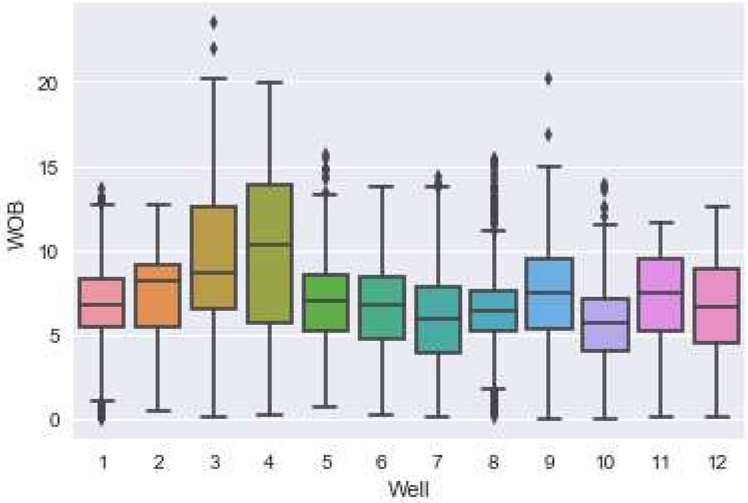
```
In [32]: sns.boxplot(x=d['Well'],y=d['ROP'])
```

```
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x22a40334748>
```



```
In [33]: D1=d[(d['WOB']>0)&(d['WOB']<25)]
In [34]: sns.boxplot(x=D1['Well'],y=D1['WOB'])
```

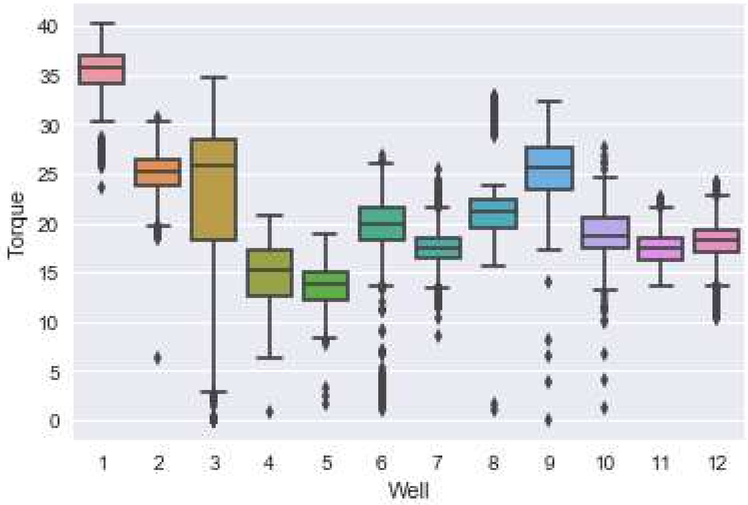
```
Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x22a404d5048>
```



```
In [35]: D2=d[(d['Torque']>0)]
```

```
In [36]: sns.boxplot(x=D2['Well'],y=D2['Torque'])
```

```
Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x22a3e9f0fd0>
```



In [126]:
```
d_s=d[['Well','Depth','GR','DT','RHOB']]
d_scut=d_s[['GR','DT','RHOB']]
```

In [127]:
```
d.shape
```

Out[127]:
```
(51877, 22)
```

In [128]:
```
from scipy import stats
d_scut=d_scut[(np.abs(stats.zscore(d_scut)) < 3).all(axis=1)]
```

In [129]:
```
d_s=d_s.merge(d_scut,on=['GR','DT','RHOB'])
```

In [130]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, init='k-means++', n_init=10, max_iter=300)
kmeans.fit(d_s[['GR','DT','RHOB']])
print(kmeans.labels_)
print(kmeans.cluster_centers_)
y_kmeans = kmeans.predict(d_s[['GR','DT','RHOB']])
```

```
[1 1 1 ... 2 2 2]
[[ 44.0424164   84.51406858   2.40045972]
 [ 11.53640007  67.03649884   2.54585863]
 [116.01160529  92.08457433   2.48356251]]
```

In [131]:
```
original_ds=d_s.copy()
```

In [132]:
```
d_s['Facies']=y_kmeans.tolist()
```

In [133]:
```
MWD1=d[['Well','Depth','ROP','WOB','Torque','SurfRPM','DownP','Mudflow','ECD'
]]
MWD1_cut=MWD1[['ROP','WOB','Torque','SurfRPM','DownP','Mudflow','ECD']]
```

In [134]:
```
from scipy import stats
MWD1_cut=MWD1_cut[(np.abs(stats.zscore(MWD1_cut)) < 3).all(axis=1)]
```

In [135]:
```
MWD1=MWD1.merge(MWD1_cut,on=['ROP','WOB','Torque','SurfRPM','DownP','Mudflow',
'ECD'])
```

In [136]:
```
d_s=d_s.merge(MWD1,on=['Well','Depth'])
```

In [137]:
```
d_s.to_csv("Wells - Copy.csv",index=False)
```

In [141]:
```
d_s = pd.read_csv("Wells - Copy.csv")
d_s.shape
```

Out[141]:
```
(72834, 13)
```

In [138]:
```
#Train_Set
d_train=d_s[d_s['Well']<10]
```

In [139]:

```
#Test_Set
d_test=d_s[d_s['Well']>=10]
```

In [140]:

```
LogData=d_s[['ROP','WOB','Torque','SurfRPM','DownP','Mudflow','ECD']].values
```

In [141]:

```
RockType=d_s['Facies'].values
```

In [117]:

```
from sklearn.preprocessing import StandardScaler
X = LogData
y = RockType
X = StandardScaler().fit_transform(X)


from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4)
```

In [142]:

```
X_train=d_train[['ROP','WOB','Torque','SurfRPM','DownP','Mudflow','ECD']].valu
es
X_test=d_test[['ROP','WOB','Torque','SurfRPM','DownP','Mudflow','ECD']].values
y_train=d_train[['Facies']].values
y_test=d_test[['Facies']].values
```

In [143]:

```
from sklearn.preprocessing import StandardScaler
X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)
```

In [144]:

```
# Import the SVM library
from sklearn.svm import NuSVC
from sklearn.svm import SVC
classifier = SVC(C = 100, kernel = 'rbf') # Define the SVM model parameters
classifier.fit(X_train, y_train) # Fit a classifier to the training data using
y-labels corresponding to properties stored in X.
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578: D
ataConversionWarning: A column-vector y was passed when a 1d array was expect
ed. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[144]:

```
SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

```
In [182]: y_pred = classifier.predict(X_test) # Predict the labels for the text X data

          # If you do a train/test split, create a confusion matrix.
          from sklearn.metrics import classification_report, confusion_matrix
          print(confusion_matrix(y_test,y_pred))
          print(classification_report(y_test,y_pred))
```

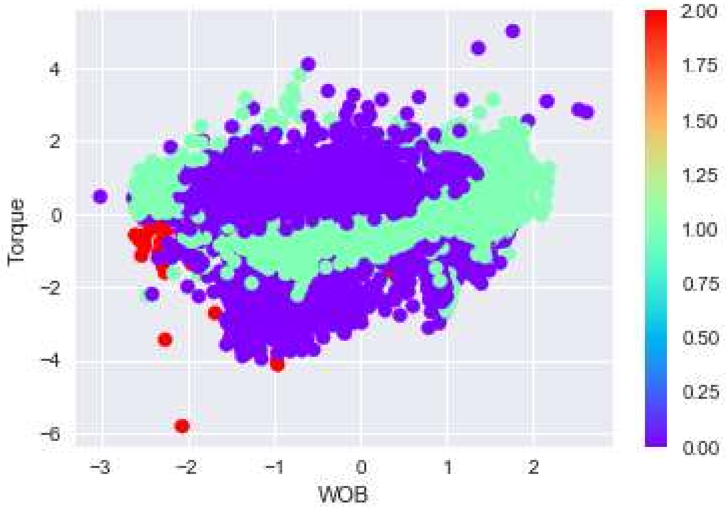
```
[[4177 2001   40]
 [ 359 3527   35]
 [ 907  148   18]]
             precision    recall  f1-score   support

          0       0.77      0.67      0.72      6218
          1       0.62      0.90      0.74      3921
          2       0.19      0.02      0.03      1073

avg / total       0.66      0.69      0.66     11212
```

```
In [146]: print('generalization performance: ', classifier.score(X_test,y_test))
```

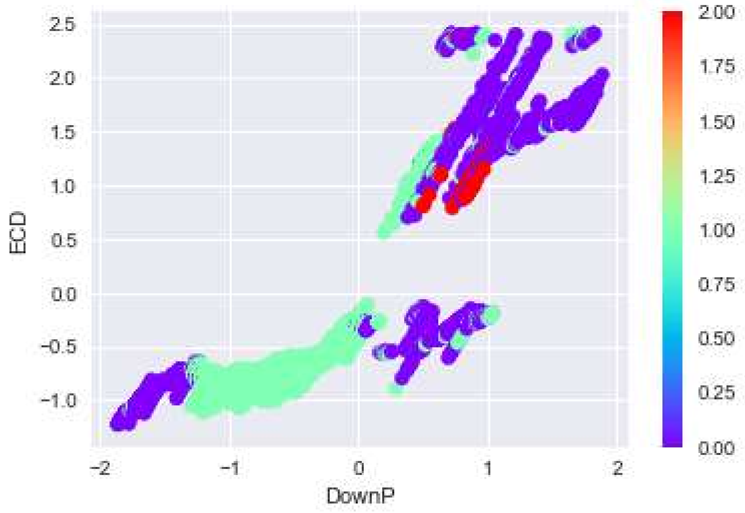
```
generalization performance:  0.6887263646093471
```

```
In [170]: plt.scatter(X_test[:,1],X_test[:,2], c=y_pred, cmap='rainbow')
          plt.xlabel('WOB')
          plt.ylabel('Torque')
          plt.colorbar()
```

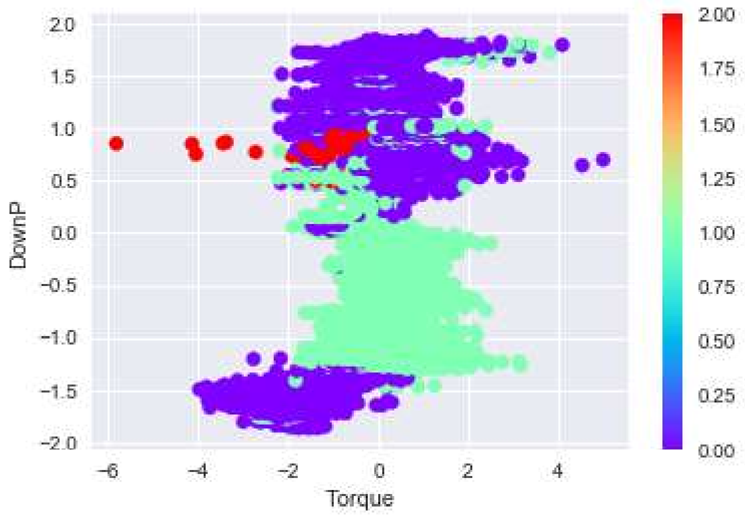
```
Out[170]: <matplotlib.colorbar.Colorbar at 0x218d39c50f0>
```



```
In [174]: plt.scatter(X_test[:,4],X_test[:,6], c=y_pred, cmap='rainbow')
          plt.xlabel('DownP')
          plt.ylabel('ECD')
          plt.colorbar()

Out[174]: <matplotlib.colorbar.Colorbar at 0x218d4e12ef0>
```



```
In [179]: plt.scatter(X_test[:,2],X_test[:,4], c=y_pred, cmap='rainbow')
          plt.xlabel('Torque')
          plt.ylabel('DownP')
          plt.colorbar()

Out[179]: <matplotlib.colorbar.Colorbar at 0x218d6f48cf8>
```



RandomForest

```
In [231]: from sklearn.feature_selection import SelectFromModel
          from sklearn.ensemble import RandomForestClassifier
          r_clf=RandomForestClassifier(n_estimators = 200, max_depth=5)
```

In [232]:
```
r_clf.fit(X_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: DataConve
rsionWarning: A column-vector y was passed when a 1d array was expected. Plea
se change the shape of y to (n_samples,), for example using ravel().
  """Entry point for launching an IPython kernel.
```

Out[232]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=5, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [233]:
```
print('memorization performance: ', r_clf.score(X_train,y_train))

print('generalization performance: ', r_clf.score(X_test,y_test))
```

```
memorization performance:  0.9106648792036282
generalization performance:  0.6073849447021049
```

In [234]:
```
y_test_pred=r_clf.predict(X_test)

from sklearn.metrics import classification_report
print(classification_report(y_test, y_test_pred))
```

```
             precision    recall  f1-score   support

          0       0.80      0.54      0.64      6218
          1       0.49      0.88      0.63      3921
          2       0.10      0.00      0.01      1073

avg / total       0.63      0.61      0.58     11212
```

In [235]:
```
feature_list=list(d_train[['ROP','WOB','Torque','SurfRPM','DownP','Mudflow','E
CD']].columns)
feature_list
```

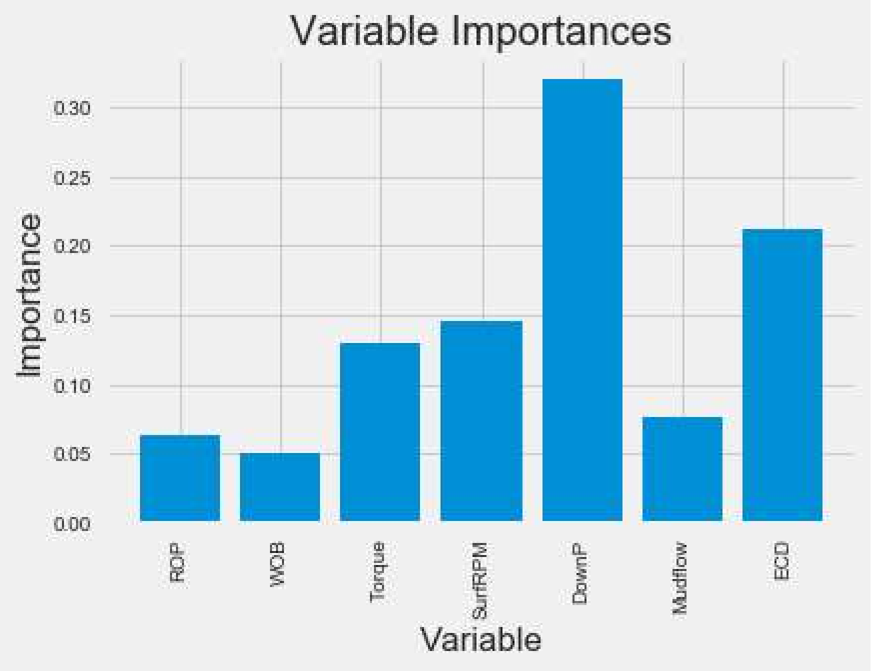
Out[235]:
```
['ROP', 'WOB', 'Torque', 'SurfRPM', 'DownP', 'Mudflow', 'ECD']
```

In [236]:

```
# Get numerical feature importances
importances = list(r_clf.feature_importances_)
# List of tuples with variable and importance
feature_importances = [(feature, round(importance, 2)) for feature, importance
in zip(feature_list, importances)]
# Sort the feature importances by most important first
feature_importances = sorted(feature_importances, key = lambda x: x[1], revers
e = True)
# Print out the feature and importances
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_imp
ortances];
```

```
Variable: DownP                Importance: 0.32
Variable: ECD                  Importance: 0.21
Variable: SurfRPM              Importance: 0.15
Variable: Torque               Importance: 0.13
Variable: Mudflow              Importance: 0.08
Variable: ROP                  Importance: 0.06
Variable: WOB                  Importance: 0.05
```

In [237]:
```
import matplotlib.pyplot as plt
%matplotlib inline
# Set the style
plt.style.use('fivethirtyeight')
# list of x locations for plotting

x_values = list(range(len(importances)))

# Make a bar chart
plt.bar(x_values, importances, orientation = 'vertical')
# Tick labels for x axis

plt.xticks(x_values, feature_list, rotation='vertical')
# Axis labels and title

plt.ylabel('Importance'); plt.xlabel('Variable'); plt.title('Variable Importan
ces');
```



In [238]:
```
from sklearn.model_selection import KFold

cv = KFold(n_splits=3, shuffle = True, random_state=125)
```

In [239]:
```
from sklearn.model_selection import GridSearchCV

param_grid = {'max_depth': [3,4,5,6,7,8], 'min_samples_split': [5,10,15]} # di
ctionary with keys

grid = GridSearchCV(r_clf, param_grid=param_grid, cv=cv, verbose=3)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:458: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)

[CV]  max_depth=8, min_samples_split=5, score=0.9366763648810542, total=  10.
8s
[CV] max_depth=8, min_samples_split=10 ...............................

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:458: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)

[CV]  max_depth=8, min_samples_split=10, score=0.9309489176007529, total=
9.2s
[CV] max_depth=8, min_samples_split=10 ...............................

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:458: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)

[CV]  max_depth=8, min_samples_split=10, score=0.9358206400821496, total=  1
0.2s
[CV] max_depth=8, min_samples_split=10 ...............................

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:458: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)

[CV]  max_depth=8, min_samples_split=10, score=0.9358206400821496, total=
9.1s
[CV] max_depth=8, min_samples_split=15 ...............................

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:458: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)

[CV]  max_depth=8, min_samples_split=15, score=0.9314623085479593, total=
8.7s
[CV] max_depth=8, min_samples_split=15 ...............................

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:458: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)

[CV]  max_depth=8, min_samples_split=15, score=0.936248502481602, total=   8.
5s
[CV] max_depth=8, min_samples_split=15 ...............................
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_validatio
n.py:458: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  estimator.fit(X_train, y_train, **fit_params)

[CV]  max_depth=8, min_samples_split=15, score=0.9365052199212733, total=
8.4s

[Parallel(n_jobs=1)]: Done  54 out of  54 | elapsed:  6.9min finished
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\model_selection\_search.p
y:739: DataConversionWarning: A column-vector y was passed when a 1d array wa
s expected. Please change the shape of y to (n_samples,), for example using r
avel().
  self.best_estimator_.fit(X, y, **fit_params)
```

```
Out[240]: GridSearchCV(cv=KFold(n_splits=3, random_state=125, shuffle=True),
                 error_score='raise',
                 estimator=RandomForestClassifier(bootstrap=True, class_weight=None, cr
          iterion='gini',
                      max_depth=5, max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=1,
                      oob_score=False, random_state=None, verbose=0,
                      warm_start=False),
                 fit_params=None, iid=True, n_jobs=1,
                 param_grid={'max_depth': [3, 4, 5, 6, 7, 8], 'min_samples_split': [5,
          10, 15]},
                 pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                 scoring=None, verbose=3)
```

```
In [241]: print(grid.best_params_)
          print("score", grid.score(X_test,y_test))
```

```
{'max_depth': 8, 'min_samples_split': 5}
score 0.6196039957188726
```

```
In [242]: print(grid.cv_results_.keys()) #for each parameter combination following metri
          cs are stored
```

```
dict_keys(['mean_fit_time', 'std_fit_time', 'mean_score_time', 'std_score_tim
e', 'param_max_depth', 'param_min_samples_split', 'params', 'split0_test_scor
e', 'split1_test_score', 'split2_test_score', 'mean_test_score', 'std_test_sc
ore', 'rank_test_score', 'split0_train_score', 'split1_train_score', 'split2_
train_score', 'mean_train_score', 'std_train_score'])
```

In [243]:
```
import pandas as pd

cv_results = pd.DataFrame(grid.cv_results_)

# generate a subset of the table
cv_results_tiny = cv_results[['param_max_depth', 'param_min_samples_split', 'm
ean_test_score','std_test_score']]

# rank them based on test scores
cv_results_tiny.sort_values(by='mean_test_score', ascending=False).head(5)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:122:
FutureWarning: You are accessing a training score ('split0_train_score'), whi
ch will not be available by default any more in 0.21. If you need training sc
ores, please set return_train_score=True
  warnings.warn(*warn_args, **warn_kwargs)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:122:
FutureWarning: You are accessing a training score ('split1_train_score'), whi
ch will not be available by default any more in 0.21. If you need training sc
ores, please set return_train_score=True
  warnings.warn(*warn_args, **warn_kwargs)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:122:
FutureWarning: You are accessing a training score ('split2_train_score'), whi
ch will not be available by default any more in 0.21. If you need training sc
ores, please set return_train_score=True
  warnings.warn(*warn_args, **warn_kwargs)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:122:
FutureWarning: You are accessing a training score ('mean_train_score'), which
will not be available by default any more in 0.21. If you need training score
s, please set return_train_score=True
  warnings.warn(*warn_args, **warn_kwargs)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:122:
FutureWarning: You are accessing a training score ('std_train_score'), which
will not be available by default any more in 0.21. If you need training score
s, please set return_train_score=True
  warnings.warn(*warn_args, **warn_kwargs)
```

Out[243]:

	param_max_depth	param_min_samples_split	mean_test_score	std_test_score
15	8	5	0.935338	0.002915
17	8	15	0.934739	0.002319
16	8	10	0.934197	0.002297
12	7	5	0.928150	0.002570
13	7	10	0.927950	0.002480

In [244]:
```
grid_new=grid.best_estimator_
```

In [245]:

```
print('memorization performance: ', grid_new.score(X_train,y_train)) # memoriz
ation

print('generalization performance: ', grid_new.score(X_test,y_test))  # genera
lization
```

```
memorization performance:  0.9419264668130866
generalization performance:  0.6196039957188726
```

PCA

In [326]:

```
d_pca=d_s[['GR','DT','RHOB']]
```

In [327]:

```
dpca=d_pca
```

In [328]:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, init='k-means++', n_init=10, max_iter=300)
kmeans.fit(dpca)
print(kmeans.labels_)
print(kmeans.cluster_centers_)
y_kmeans = kmeans.predict(dpca)
```

```
[1 1 1 ... 2 2 2]
[[ 43.67232042  83.97561779   2.40660651]
 [ 11.32876413  66.91932523   2.54839223]
 [116.71018132  91.53289838   2.48708824]]
```

In [329]:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
```

In [330]:

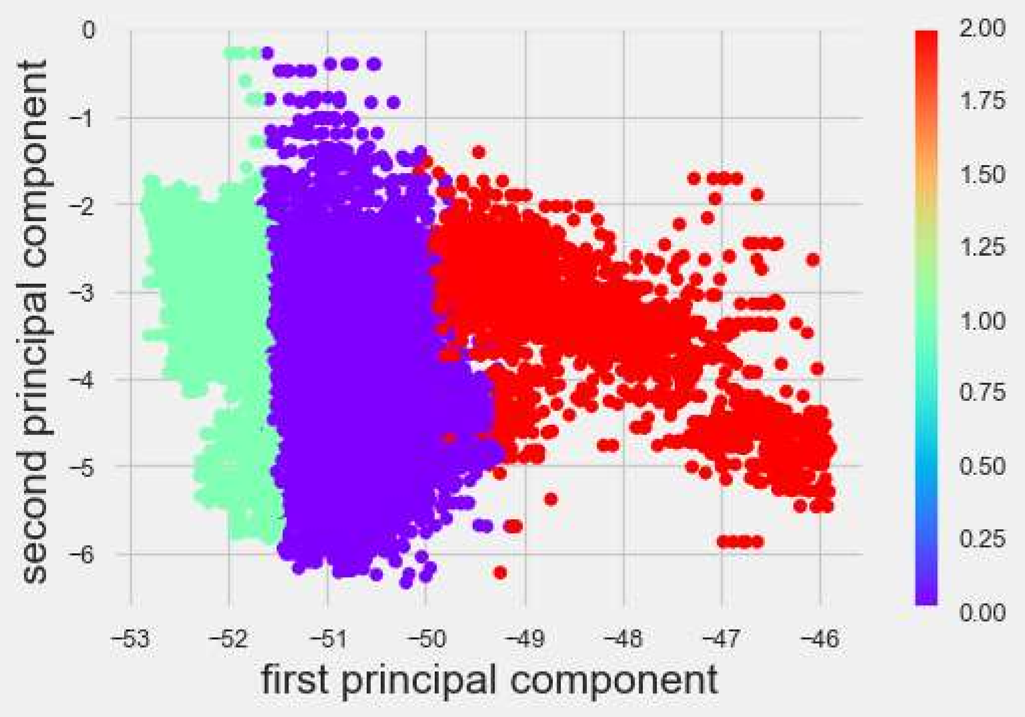
```
pca.fit(dpca)
```

Out[330]:

```
PCA(copy=True, iterated_power='auto', n_components=3, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)
```

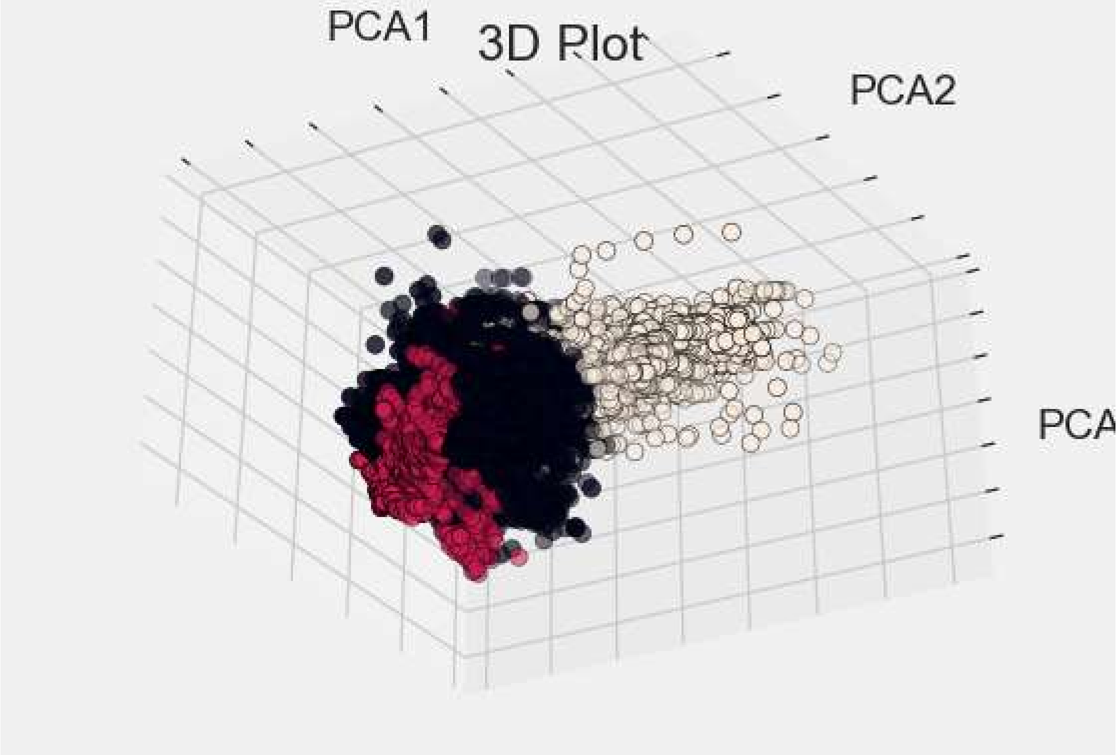
In [331]:
```
from sklearn.preprocessing import StandardScaler
X_dpca=np.array(dpca)
X_dpca = StandardScaler().fit_transform(X_dpca)
X_pca = pca.transform(X_dpca)
plt.rcParams['figure.dpi']=90
plt.scatter(X_pca[:, 0], X_pca[:, 2], linewidths=0, s=30,c=y_kmeans,cmap='rain
bow')
plt.xlabel("first principal component")
plt.ylabel("second principal component")
plt.colorbar()
```

Out[331]:
```
<matplotlib.colorbar.Colorbar at 0x218805095c0>
```



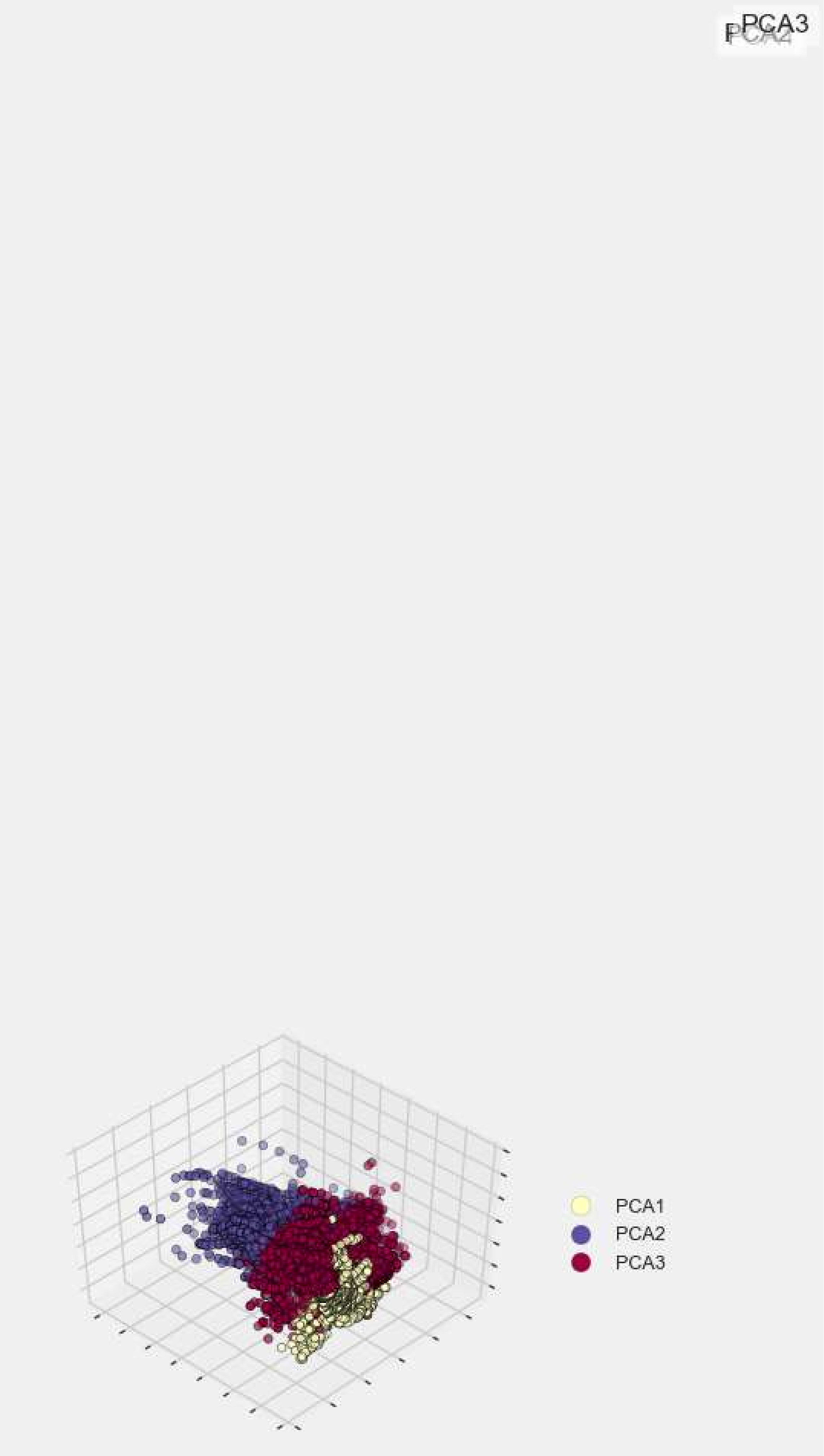
```
In [332]: from matplotlib.lines import Line2D
          import matplotlib.pyplot as plt
          from mpl_toolkits.mplot3d import Axes3D
          import numpy as np
          from sklearn import decomposition
          from sklearn import datasets
          fig = plt.figure(1)
          ax = Axes3D(fig, elev=-40, azim=300) #3D plot
          ax.scatter(X_pca[:, 0], X_pca[:, 1], X_pca[:, 2], c=y_kmeans, s = 50, edgecolo
          r='k')

          ax.w_xaxis.set_ticklabels([])
          ax.w_yaxis.set_ticklabels([])
          ax.w_zaxis.set_ticklabels([])
          ax.set_xlabel('PCA1')
          ax.set_ylabel('PCA2')
          ax.set_zlabel('PCA3')
          ax.set_title('3D Plot')
```

```
Out[332]: Text(0.5,0.92,'3D Plot')
```



In [333]:

```
fig = plt.figure(figsize=(5.5, 3))
ax = Axes3D(fig, rect=[0, 0, .7, 1], elev=48, azim=134)
labelTups = [('PCA1', 0), ('PCA2', 1), ('PCA3', 2)]
for name, label in labelTups:
    ax.text3D(X[y == label, 0].mean(),
              X[y == label, 1].mean() + 1.5,
              X[y == label, 2].mean(), name,
              horizontalalignment='center',
              bbox=dict(alpha=.5, edgecolor='w', facecolor='w'))
# Reorder the labels to have colors matching the cluster results

sc = ax.scatter(X_pca[:, 0], X_pca[:, 1], X_pca[:, 2], c=y_kmeans, cmap="Spect
ral", edgecolor='k')

ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])

colors = [sc.cmap(sc.norm(i)) for i in [1, 2, 0]]
custom_lines = [plt.Line2D([],[], ls="", marker='.',
                mec='k', mfc=c, mew=.1, ms=20) for c in colors]
ax.legend(custom_lines, [lt[0] for lt in labelTups],
          loc='center left', bbox_to_anchor=(1.0, .5))

plt.show()
```


PCA3
PCA2
PCA1
PCA2
PCA3