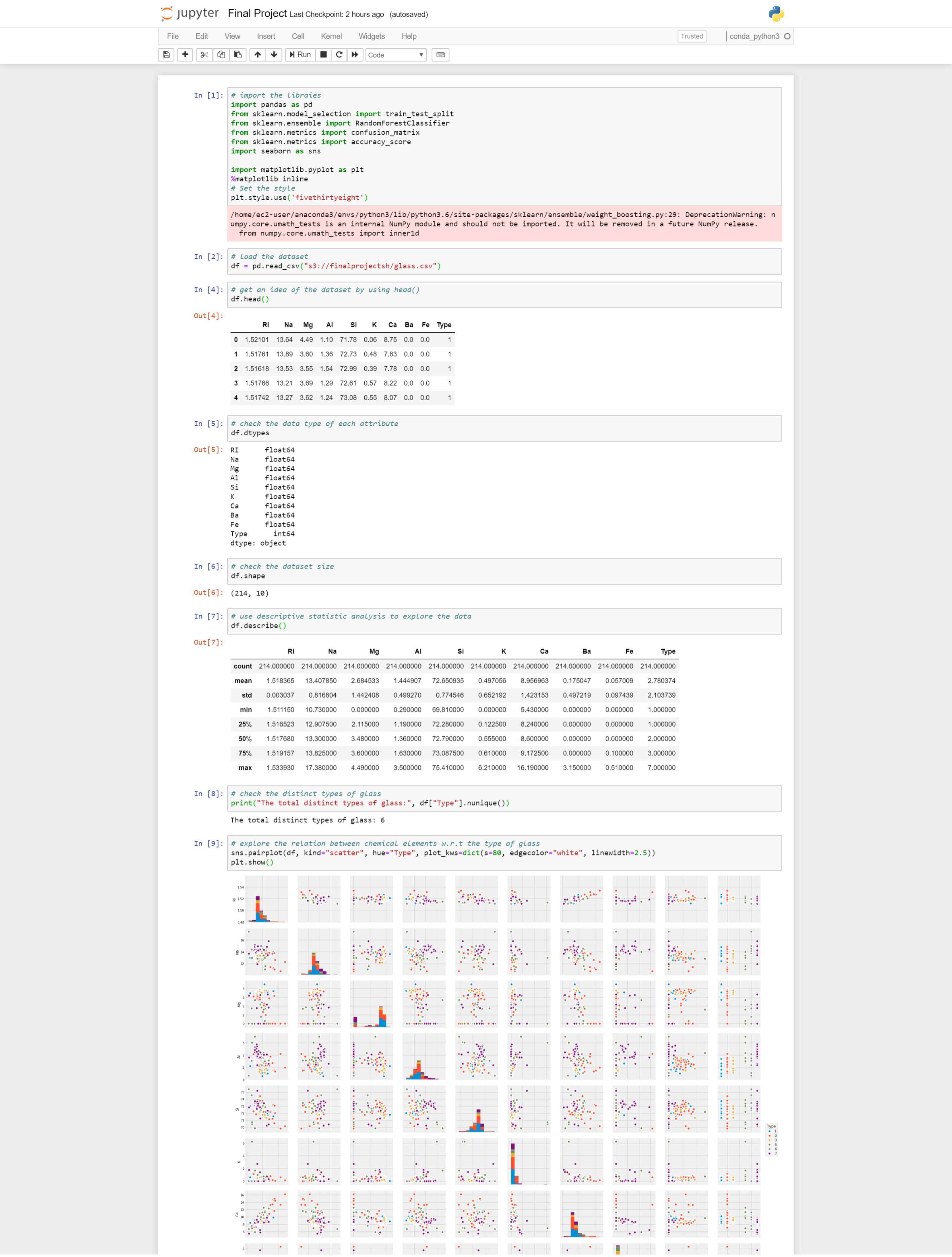


Final Project Last Checkpoint: 2 hours ago (autosaved)

In [1]:
```
# import the Libraies
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
import seaborn as sns

import matplotlib.pyplot as plt
%matplotlib inline
# Set the style
plt.style.use('fivethirtyeight')
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarning: n
umpy.core.umath_tests is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [2]:
```
# Load the dataset
df = pd.read_csv("s3://finalprojectsh/glass.csv")
```

In [4]:
```
# get an idea of the dataset by using head()
df.head()
```

Out[4]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

In [5]:
```
# check the data type of each attribute
df.dtypes
```

Out[5]:
```
RI      float64
Na      float64
Mg      float64
Al      float64
Si      float64
K       float64
Ca      float64
Ba      float64
Fe      float64
Type      int64
dtype: object
```

In [6]:
```
# check the dataset size
df.shape
```

Out[6]:
```
(214, 10)
```

In [7]:
```
# use descriptive statistic analysis to explore the data
df.describe()
```

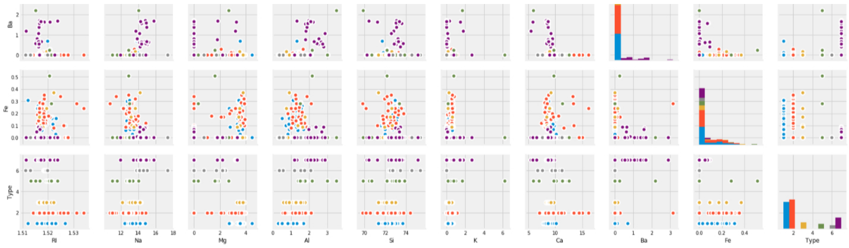
Out[7]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	0.175047	0.057009	2.780374
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	0.497219	0.097439	2.103739
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.000000	0.000000	1.000000
25%	1.516523	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	0.000000	0.000000	1.000000
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	0.000000	0.000000	2.000000
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	0.000000	0.100000	3.000000
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	3.150000	0.510000	7.000000

In [8]:
```
# check the distinct types of glass
print("The total distinct types of glass:", df["Type"].nunique())
```

```
The total distinct types of glass: 6
```

In [9]:
```
# explore the relation between chemical elements w.r.t the type of glass
sns.pairplot(df, kind="scatter", hue="Type", plot_kws=dict(s=80, edgecolor="white", linewidth=2.5))
plt.show()
```



In [10]:
```
# set up the features and label
y = df["Type"]
X = df.drop("Type", axis = 1)
```

In [11]:
```
# split the training data and test data
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size = 0.3, random_state = 0)
print('Training Features Shape:', train_X.shape)
print('Training Label Shape:', train_y.shape)
print('Testing Features Shape:', test_X.shape)
print('Testing Label Shape:', test_y.shape)
```

```
Training Features Shape: (149, 9)
Training Label Shape: (149,)
Testing Features Shape: (65, 9)
Testing Label Shape: (65,)
```

In [12]:
```
rf = RandomForestClassifier(n_estimators=20,random_state=0)
rf.fit(train_X, train_y)
y_pred = rf.predict(test_X)
print("Confusion Matrix:", confusion_matrix(test_y, y_pred), sep="\n")
print("\n Accuracy Score:", round(accuracy_score(test_y,y_pred)*100, 2), '%.')
```

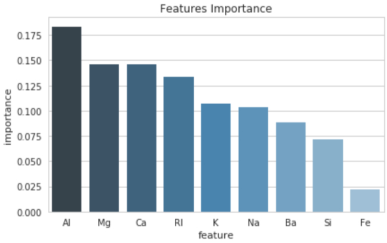
```
Confusion Matrix:
[[17  4  0  0  0  0]
 [ 8 16  0  1  0  1]
 [ 5  0  2  0  0  0]
 [ 0  0  0  2  0  0]
 [ 0  0  0  0  2  0]
 [ 0  0  0  0  0  7]]

 Accuracy Score: 70.77 %.
```

In [13]:
```
# create an importance Matrix
features = pd.DataFrame()
features['feature'] = X.columns
features['importance'] = rf.feature_importances_
features.sort_values(by=['importance'], ascending = False, inplace = True)
print("Feature importance Matix:",features, sep = "\n")
```

```
Feature importance Matix:
  feature  importance
3      Al    0.183422
2      Mg    0.146107
6      Ca    0.145945
0      RI    0.133366
5       K    0.106585
1      Na    0.103140
7      Ba    0.088586
4      Si    0.071159
8      Fe    0.021690
```

In [14]:
```
# visualize the feature importance
sns.set(style="whitegrid")
ax = sns.barplot(x="feature", y="importance", data=features, palette="Blues_d")
ax.set_title('Features Importance')
plt.show()
```



In [ ]: