3803ICT
Big Data Analysis

Lab 03 – Exploratory Data Analysis

Trimester 1 - 2019

Table of Contents

I. Visualization with Matplotlib

1. Example

Create new jupyter notebook and follow below steps to visualize using Matplotlib

In [1]:
```
import matplotlib.pyplot as plt
```

In [3]:
```
import numpy as np
x = np.linspace(0, 5, 11)
y = x ** 2
```

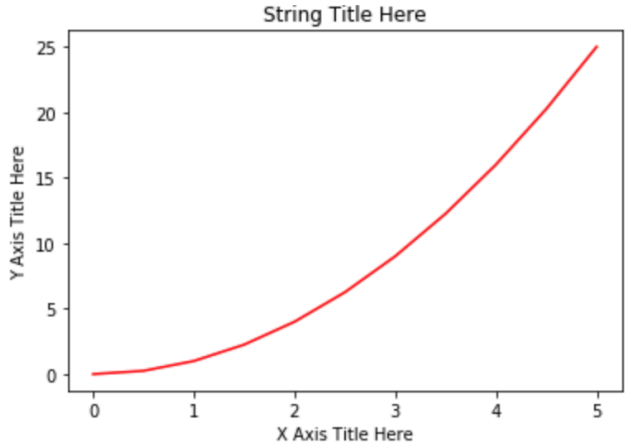
In [4]:
```
x
```

Out[4]:
```
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. ])
```

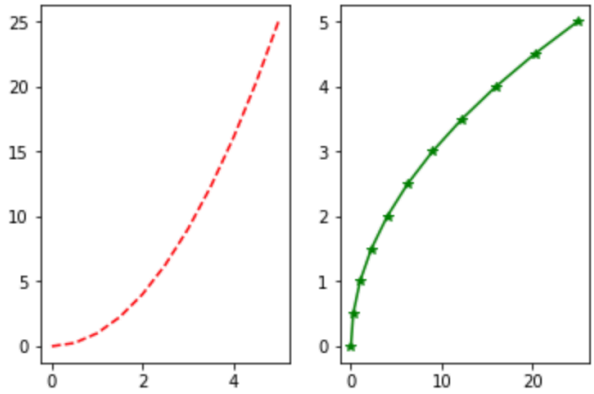
In [5]:
```
y
```

Out[5]:
```
array([ 0.  ,  0.25,  1.  ,  2.25,  4.  ,  6.25,  9.  , 12.25, 16.  ,
       20.25, 25.  ])
```

In [6]:
```
plt.plot(x, y, 'r') # 'r' is the color red
plt.xlabel('X Axis Title Here')
plt.ylabel('Y Axis Title Here')
plt.title('String Title Here')
plt.show()
```



In [7]:
```
# plt.subplot(nrows, ncols, plot_number)
plt.subplot(1,2,1)
plt.plot(x, y, 'r--') # More on color options later
plt.subplot(1,2,2)
plt.plot(y, x, 'g*-');
```



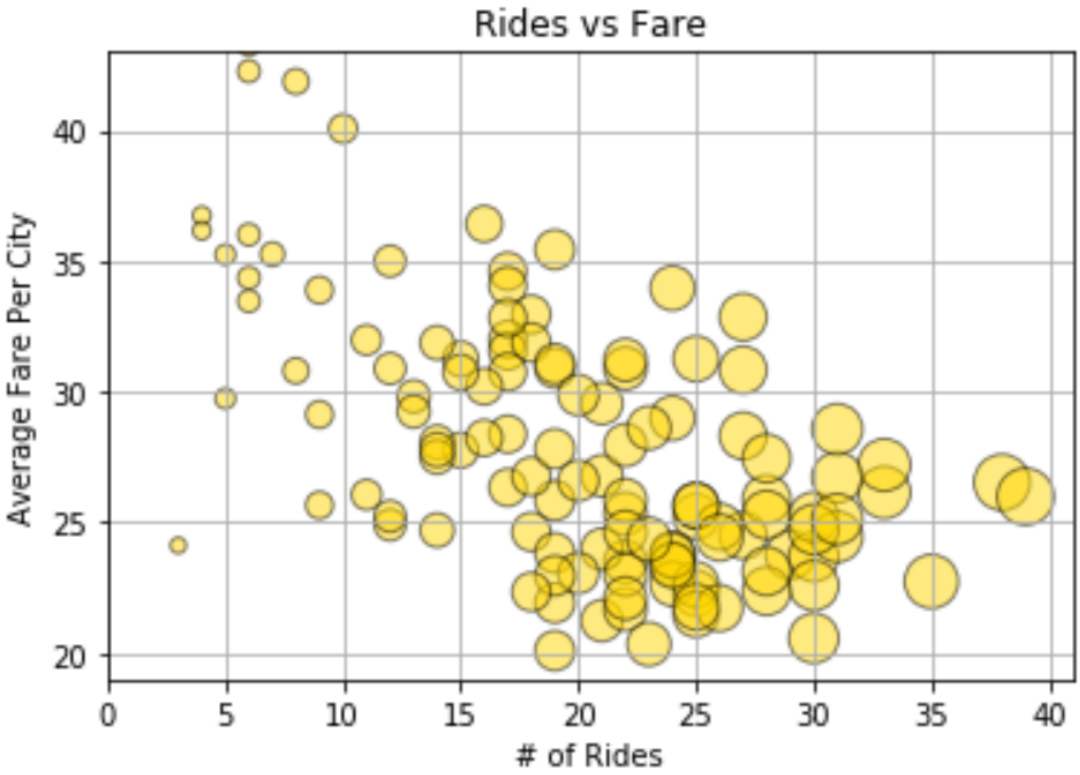
2. Exercises

Create new Jupiter notebook and implement following things:

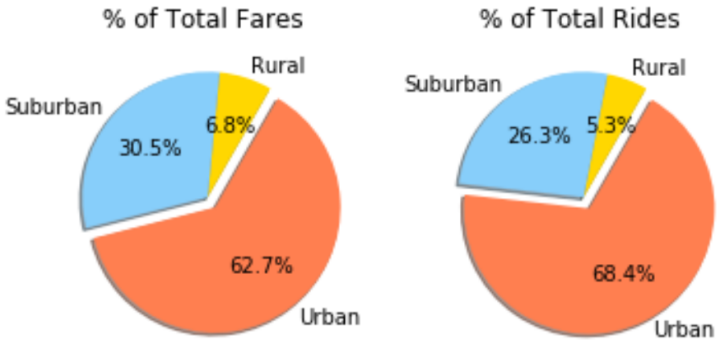
- Use pandas to load city_data.csv and ride_data.csv
- Merge them into 1 DataFrame by city.
- Calculate and create a new DataFrame like follow image:

city	Average Fare	Number of Drivers	Number of Rides
Amandaburgh	24.641667	18	18
Barajasview	25.332273	22	22
Barronchester	36.422500	16	16
Bethanyland	32.956111	18	18
Bradshawfurt	40.064000	10	10

- Draw bubble plot for above DataFrame (hint: use plot scatter)



- Draw % of Total Fares and Rides in pie chart (hint: use subplot)



II. Visualisation with Seaborn

1. Example

In [2]:
```
import seaborn as sns
%matplotlib inline
```

In [3]:
```
tips = sns.load_dataset('tips')
```

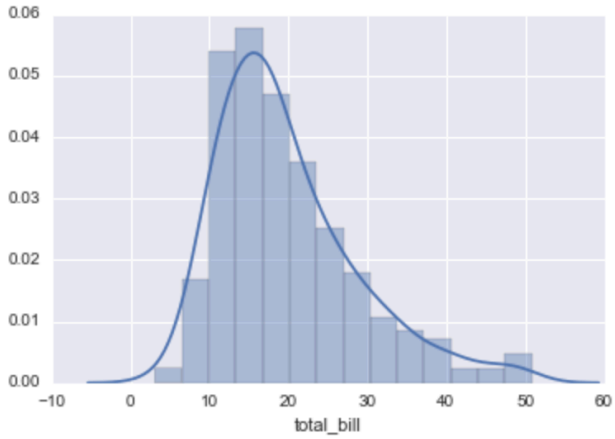
In [4]:
```
tips.head()
```

Out[4]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

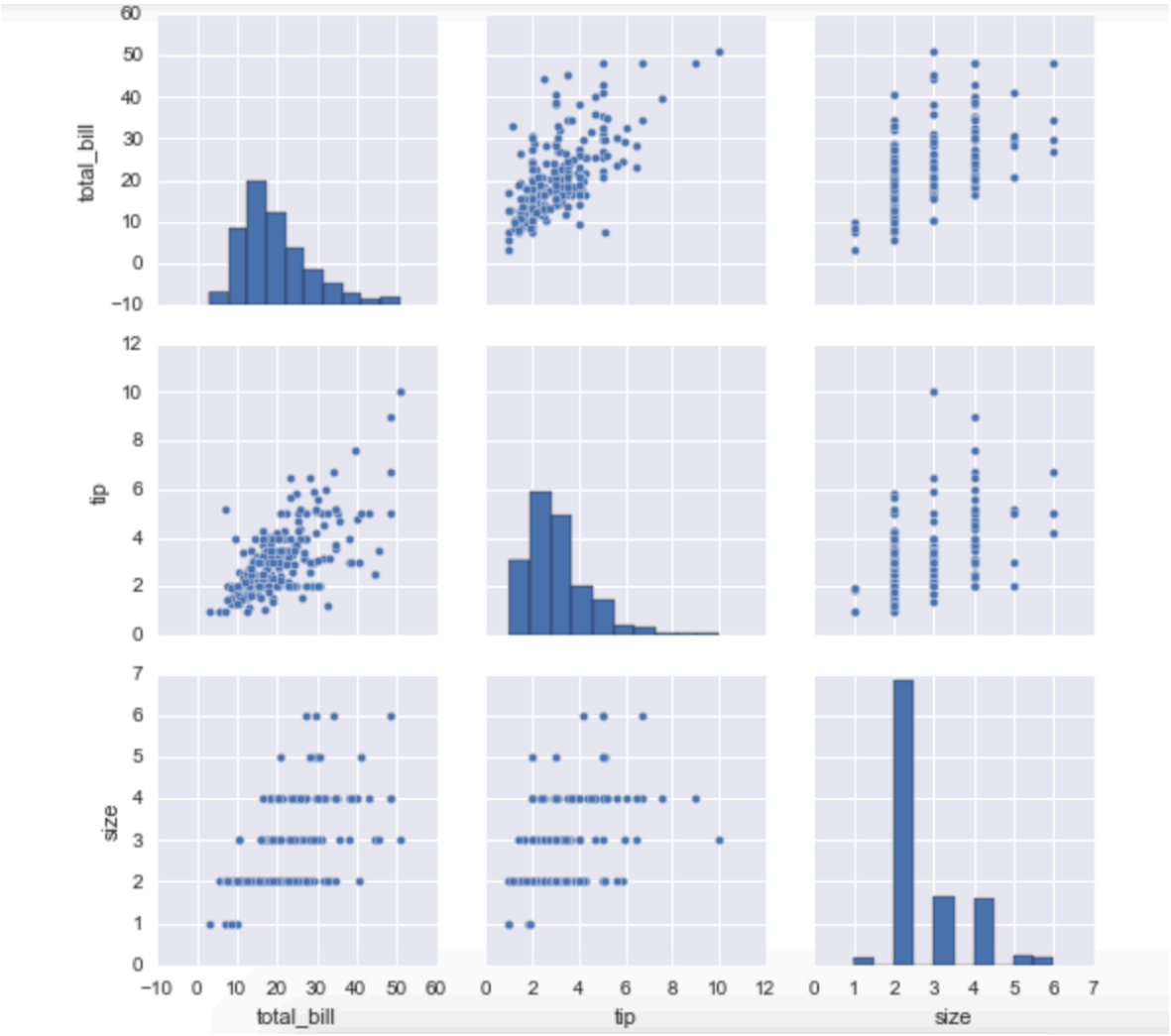
In [16]:
```
sns.distplot(tips['total_bill'])
# Safe to ignore warnings
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11dd8e5f8>
```



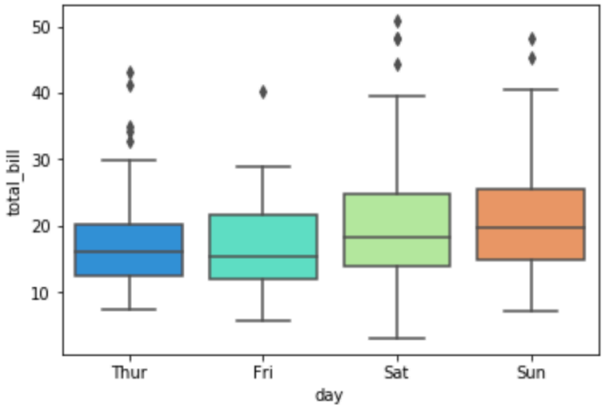
In [18]:
```
sns.pairplot(tips)
```

Out[18]:
```
<seaborn.axisgrid.PairGrid at 0x11e844208>
```

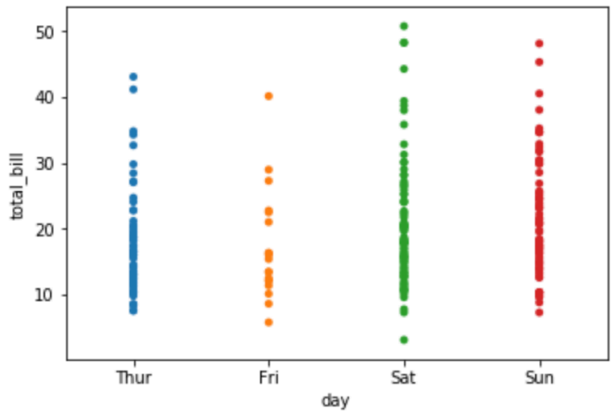


In [6]:
```
sns.boxplot(x="day", y="total_bill", data=tips,palette='rainbow')
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x111b93630>



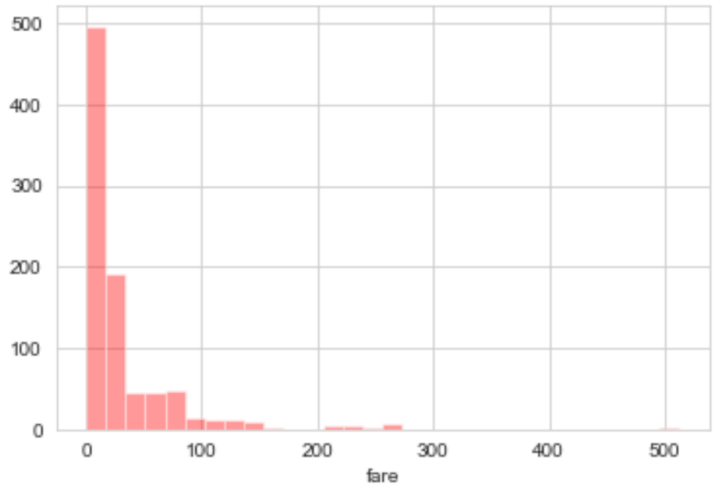
```
In [7]: sns.stripplot(x="day", y="total_bill", data=tips)
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x111bb9630>
```



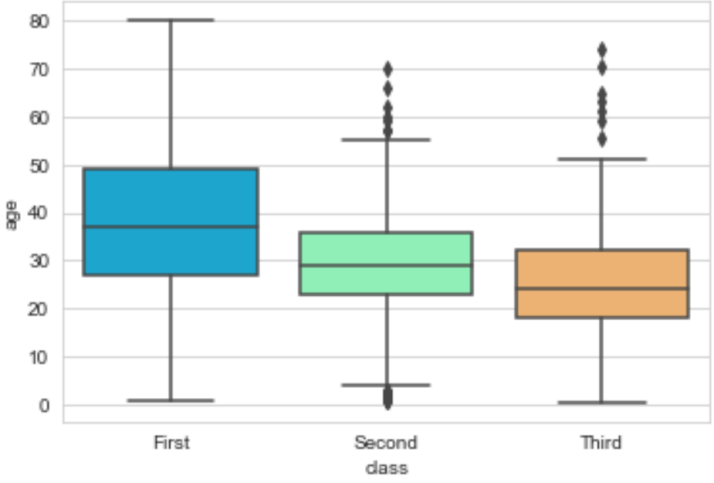
2. Exercises

Create new Jupiter notebook and implement following things:

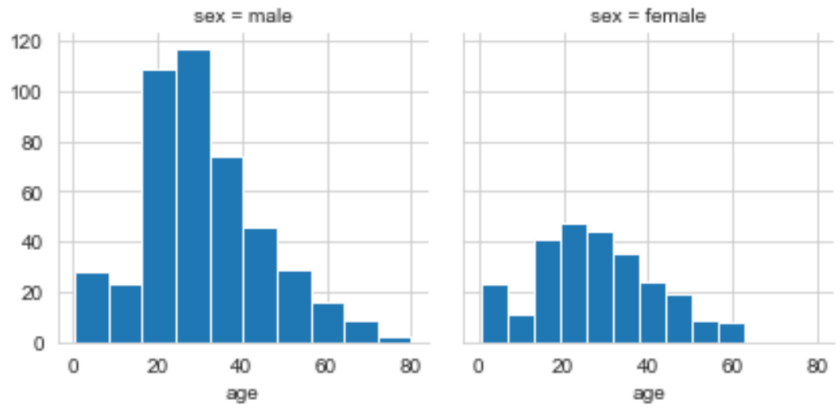
- Load titanic dataset with seaborn (sns.load_dataset('titanic')).
- Create a barchart plot for “fare”.



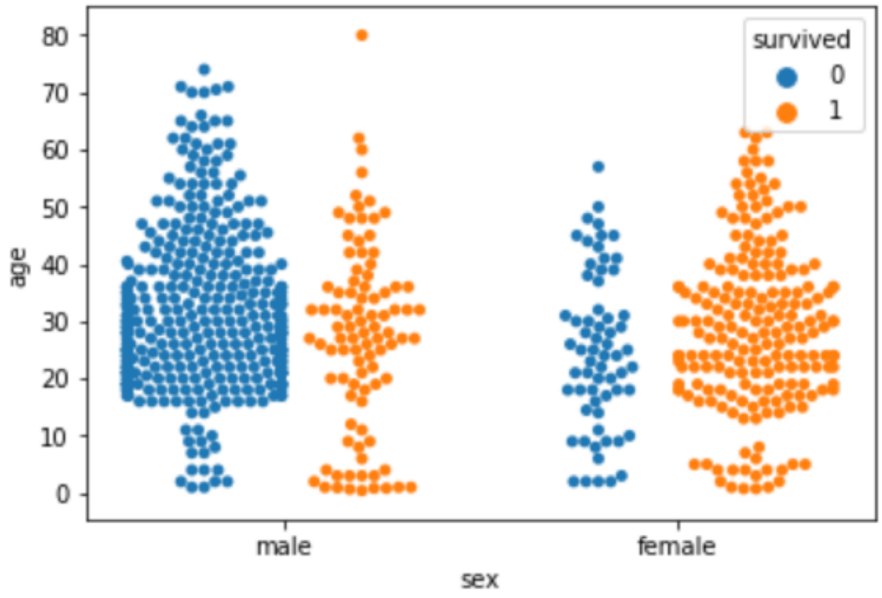
- Create a boxplot between “age” and “class”.



- Statistic and compare between age and sex.



- Use swarmplot to draw following chart.



III. Dimensionality Reduction

1. Principal Component Analysis (PCA)

Apply PCA to reduce the dimension of Iris dataset (the dataset has 150 samples individual flowers and three distinct Iris species that each flower is classified into).

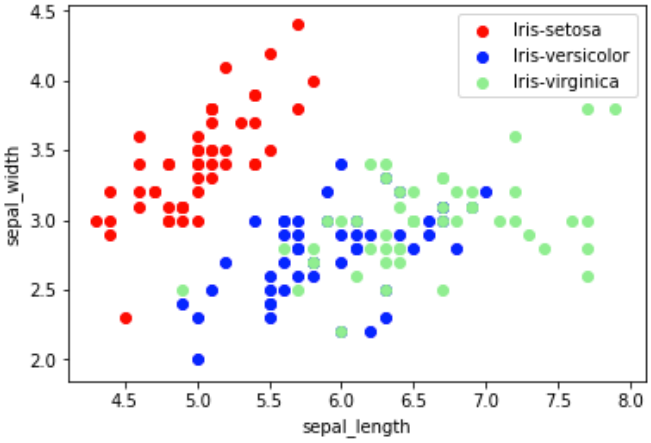
❖ Load the dataset

```
In [68]: import matplotlib.pyplot as plt
         import pandas as pd

         from sklearn.decomposition import PCA as sklearnPCA
```

```
In [69]: url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
         data = pd.read_csv(url,header=None)

         y = data[4] # Split off classifications
         X = data.iloc[:,0:4] # Split off features
```

❖ Visualize the dataset into scatter series.

In [75]:
```
# three different scatter series so the class labels in the legend are distinct
plt.scatter(X[y=='Iris-setosa'].iloc[:,0], X[y=='Iris-setosa'].iloc[:,1], label='Iris-setosa', c='red')
plt.scatter(X[y=='Iris-versicolor'].iloc[:,0], X[y=='Iris-versicolor'].iloc[:,1], label='Iris-versicolor', c='blue')
plt.scatter(X[y=='Iris-virginica'].iloc[:,0], X[y=='Iris-virginica'].iloc[:,1], label='Iris-virginica', c='lightgreen')

# Prettify the graph
plt.legend()
plt.xlabel('sepal_length')
plt.ylabel('sepal_width')

# display
plt.show()
```

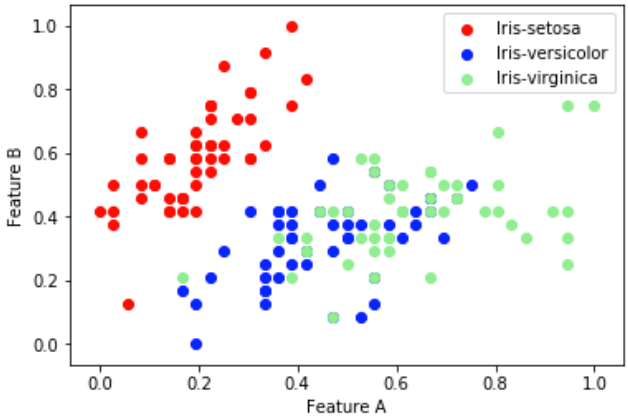


- Rescale the data to a [0,1] range.

In [71]:
```
X_norm = (X - X.min())/(X.max() - X.min())
```

- Plot the X_norm again:

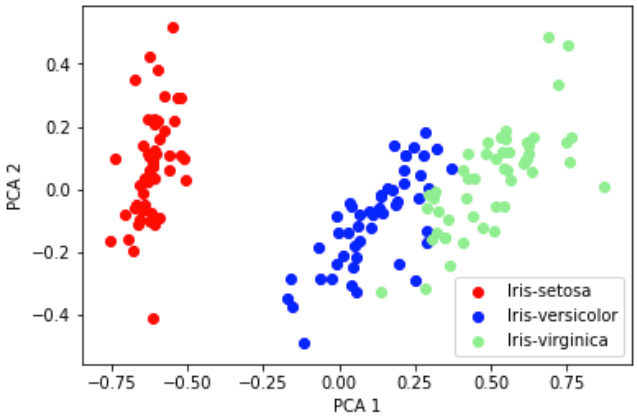
In [72]:
```
# three different scatter series so the class labels in the legend are distinct
plt.scatter(X_norm[y=='Iris-setosa'].iloc[:,0], X_norm[y=='Iris-setosa'].iloc[:,1], label='Iris-setosa', c='red')
plt.scatter(X_norm[y=='Iris-versicolor'].iloc[:,0], X_norm[y=='Iris-versicolor'].iloc[:,1], label='Iris-versicolor', c='blue')
plt.scatter(X_norm[y=='Iris-virginica'].iloc[:,0], X_norm[y=='Iris-virginica'].iloc[:,1], label='Iris-virginica', c='lightgreen')

# Prettify the graph
plt.legend()
plt.xlabel('Feature A')
plt.ylabel('Feature B')

# display
plt.show()
```



- Use sklearn PCA object to the normalized dataset and visualize it again in the plot.

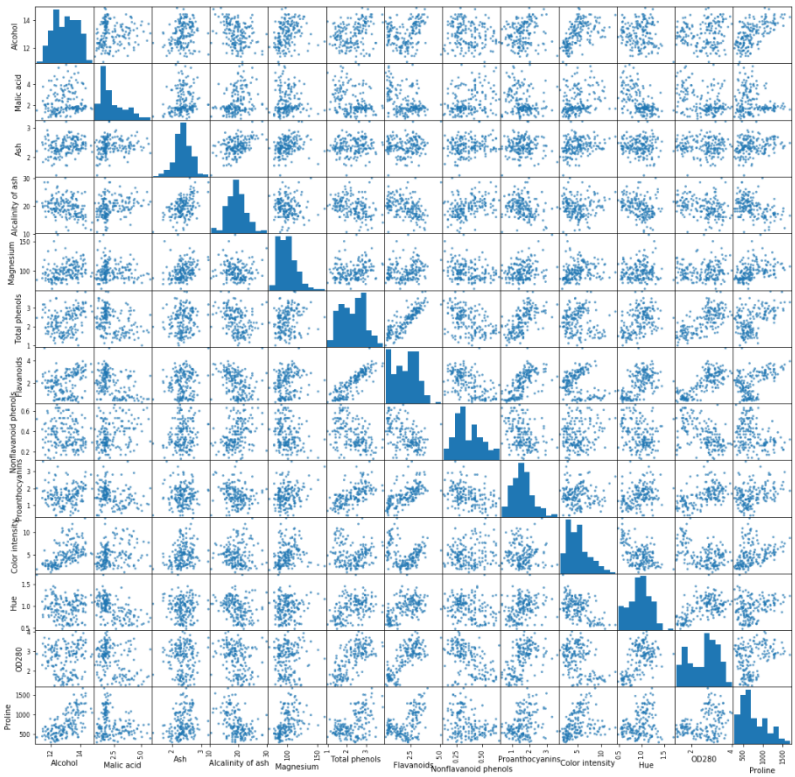
In [73]:
```
pca = sklearnPCA(n_components=2) #2-dimensional PCA
transformed = pd.DataFrame(pca.fit_transform(X_norm))
```

In [74]:
```
# three different scatter series so the class labels in the legend are distinct
plt.scatter(transformed[y=='Iris-setosa'].iloc[:,0], transformed[y=='Iris-setosa'].iloc[:,1], label='Iris-setosa', c='red')
plt.scatter(transformed[y=='Iris-versicolor'].iloc[:,0],
            transformed[y=='Iris-versicolor'].iloc[:,1], label='Iris-versicolor', c='blue')
plt.scatter(transformed[y=='Iris-virginica'].iloc[:,0],
            transformed[y=='Iris-virginica'].iloc[:,1], label='Iris-virginica', c='lightgreen')

# Prettify the graph
plt.legend()
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')

# display
plt.show()
```

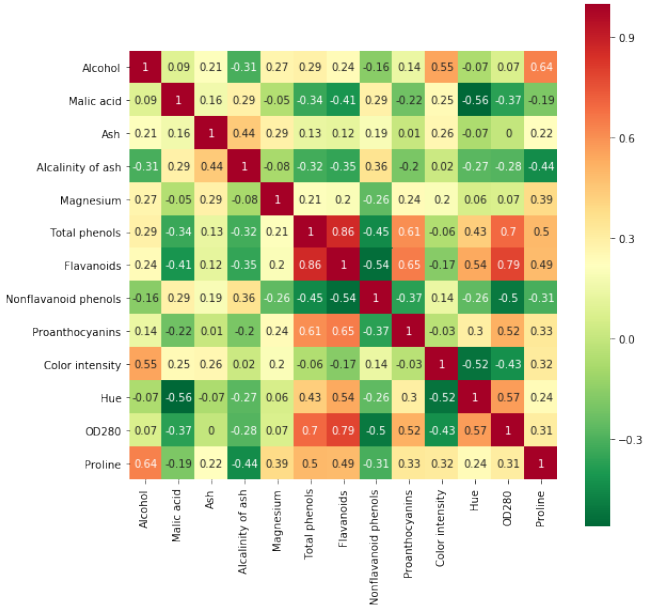


2. Exercise

- Load data from wine.data.csv file. Keep 1st column into a separate variable (label) and remove it from DataFrame.
- Use Scatter plot to learn attributes of data. What is your conclusion?



❖ Try to visualize data with correlation heatmap? Can you find any pairs of attributes which have large correlation?



❖ Normalize data by removing the mean and scaling to unit variance using `preprocessing.StandardScaler`.

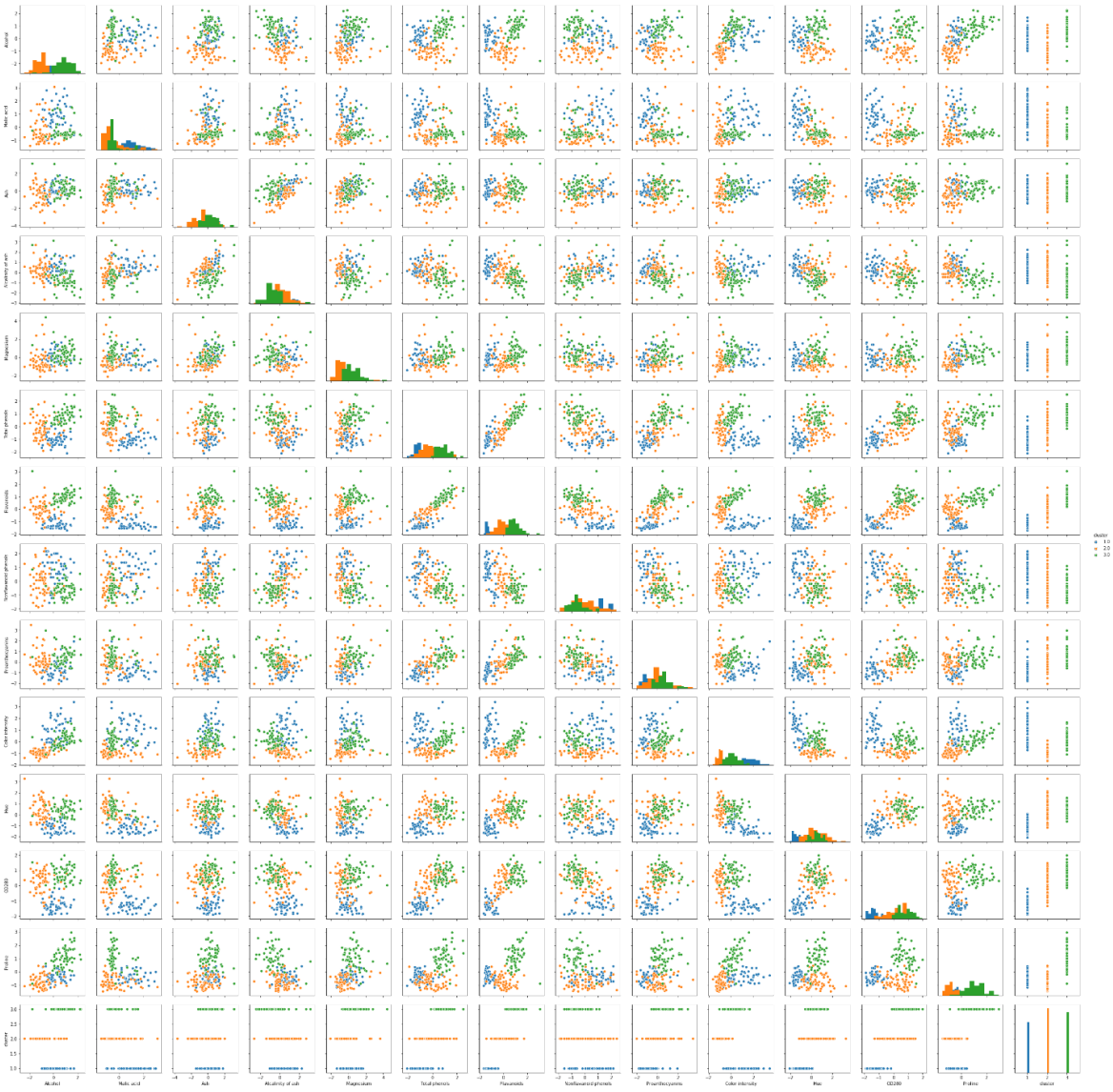
Hint:
standardScaler = preprocessing.StandardScaler()
standardScaler.fit(wine)
X_scaled_array = standardScaler.transform(wine)
normalizedData = pd.DataFrame(X_scaled_array, columns = wine.columns)

❖ Use kMeans to cluster the normalized data (By using Elbow method https://en.wikipedia.org/wiki/Elbow_method_(clustering) , the number of clusters should be 3). Use pairplot to visualize the wine attributes with their cluster.

Hint:
kMeansClustering = KMeans(n_clusters = 3, random_state=seed)
res = kMeansClustering.fit_predict(normalizedData)

---

normalizedData ["cluster"] = label_pred_KM.astype('float64')
sns_plot = sns.pairplot(normalizedData, hue = "cluster",diag_kind="hist")



- By using explained_variance_ratio_ attribute, we know that first 6 PCs explain 85.1% of variance if we reduce the dimension using PCA. You need to apply PCA with 6 components for the above normalized data. Then applying kMeans (3 clusters) to cluster the data after dimensionality reduction.
- Use adjusted_rand_score in sklearn.metrics.cluster to calculate the scores of original kMeans and kMeans after PCA. What is your conclusion?

  Hint:
  adjusted_rand_score(label, label_pred_KM_PCA)
  with label is kept in 1st step.