University of Canberra
Faculty of Science and Technology

11482 Pattern Recognition and Machine Learning
11512 Pattern Recognition and Machine Learning PG

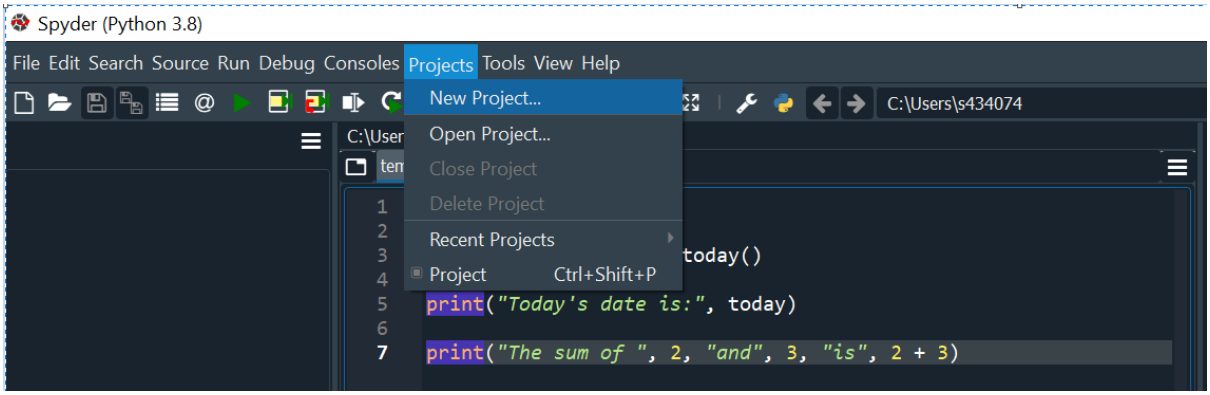
Week 3 Tutorial
Understanding Concepts, Data Visualization and Linear regression

Objectives

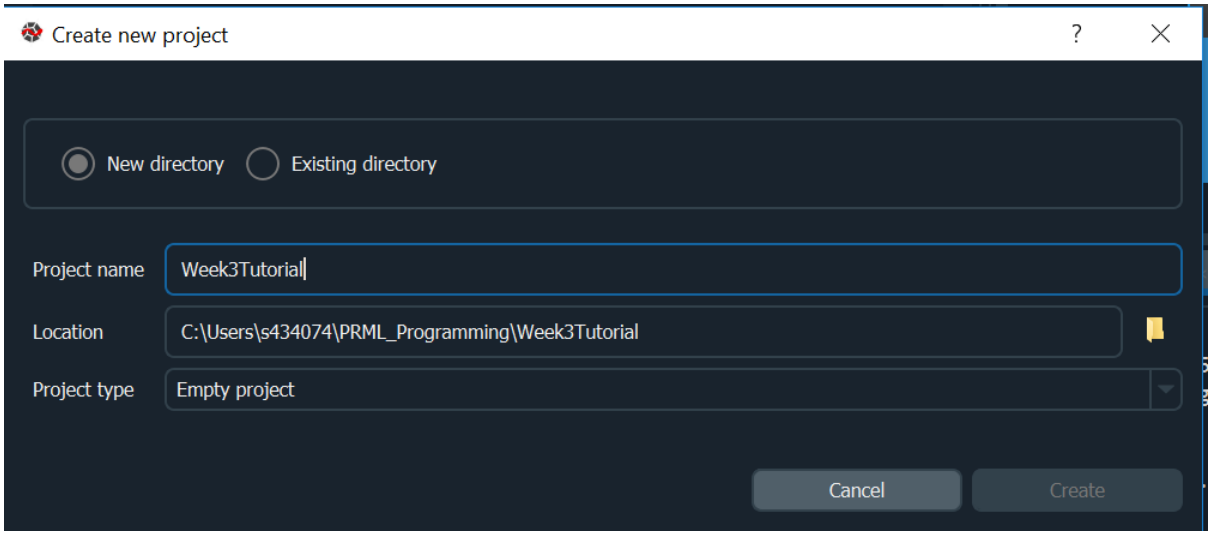
- To use Spyder to create a new Python project
- To practise read a data and understand features , describe data
- To practise data visualization using Matplotlib
- To practise a linear regression model , analyse model fitting

Create a project in Spyder

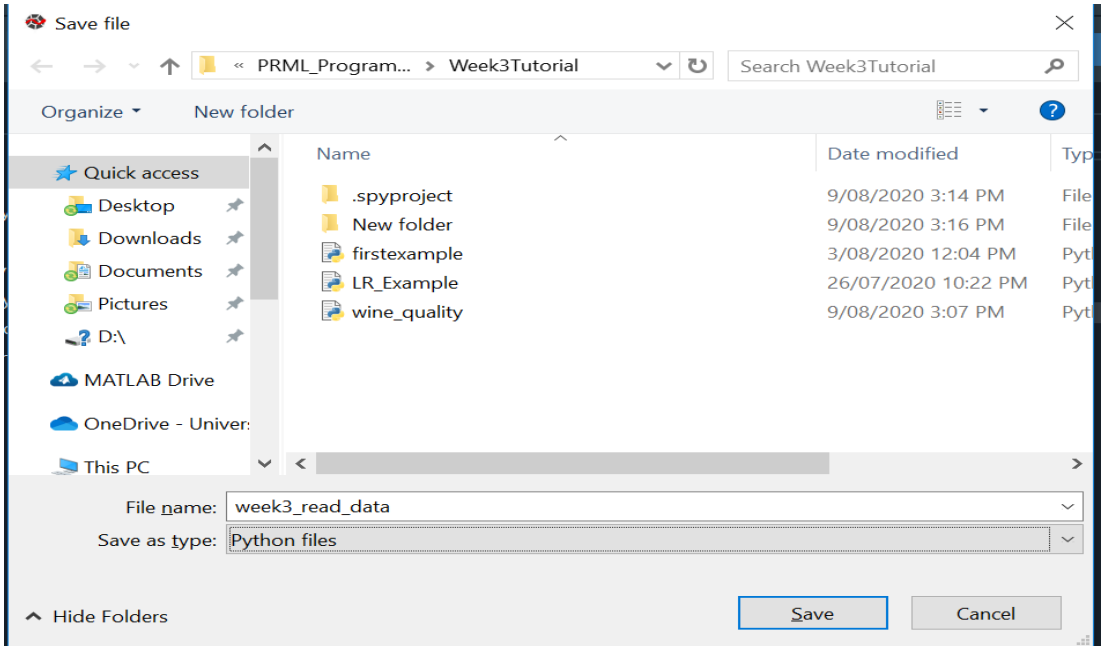
- Open Spyder. Click on Project > New Project



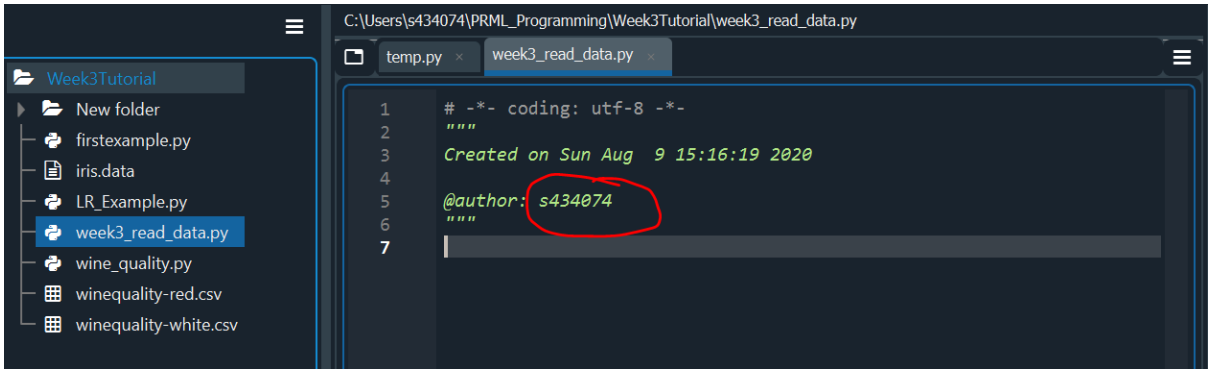
- On the New Project dialog, select **New directory**. Enter **Week3Tutorial** to the **Name** text box. Finally, click OK button to create this new project. The location is **C:\Users\s434074\PRML_Programming\Week3Tutorial**



- Then clicking on create a new file and save it as week3_read_data



- Here is an empty file after being created. You will write Python code in this file.



Practise to read a csv file

- Go to this link https://archive.ics.uci.edu/ml/datasets/iris and download an iris.data file. Then save the data to the same folder Week3Tutorial.
- Enter the following to week3_read_data.py (Note: **all lines must have the same indentation**)

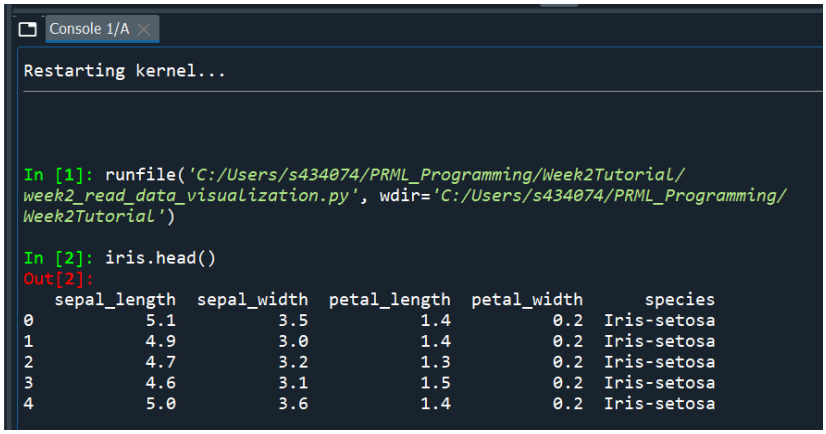
```
import pandas as pd
import matplotlib.pyplot as plt

#read iris data
iris = pd.read_csv("iris.data", sep=',', names=["sepal_length", "sepal_width", "petal_length",
                                                "petal_width", "species"])
```

The first two lines are to import two library: pandas and matplotlib.
The last line is to read the iris file as a DataFrame and save to a parameter iris with adding a row of names for each column.

Running the program: click on  button or press F5

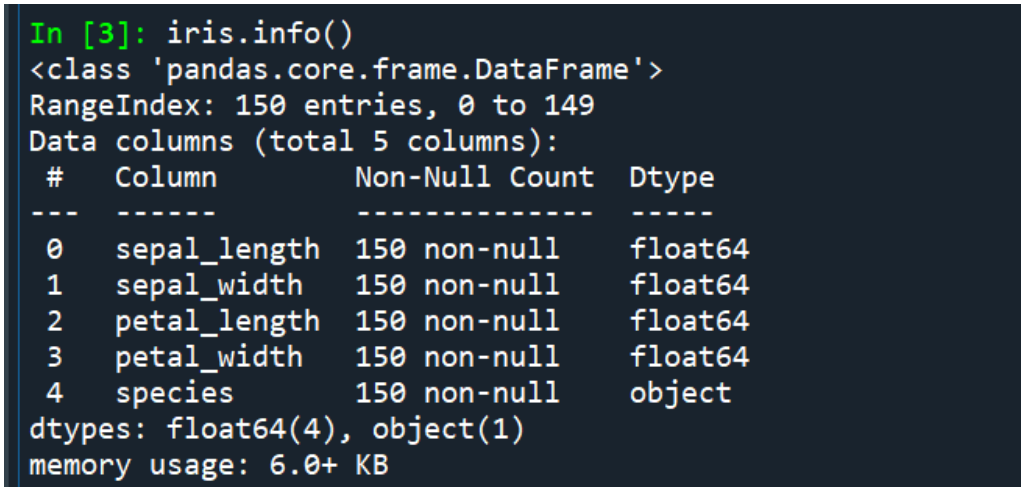
Exploring the data

- In the console, type iris.head() to see the first five rows

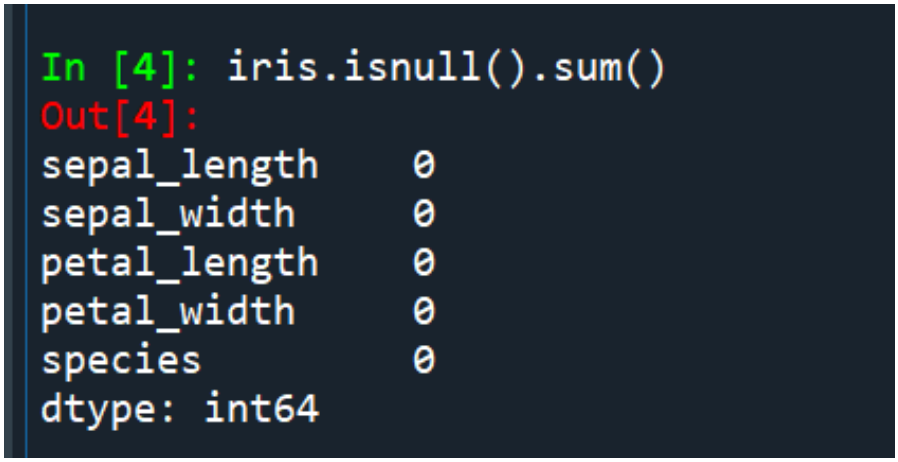


If you cannot see the Console, please select View -> Panes -> IPython Console

- To see a summary of data, enter iris.info().



- To identify whether there are any missing values or NaN, enter iris.isnull().sum()



- TO see how many attributes we have of each species, enter iris['species'].value_counts()

```
In [6]: iris['species'].value_counts()
Out[6]:
Iris-virginica     50
Iris-setosa        50
Iris-versicolor    50
Name: species, dtype: int64
```

- To see a 'species column, type iris['species']

```
In [5]: iris['species']
Out[5]:
0          Iris-setosa
1          Iris-setosa
2          Iris-setosa
3          Iris-setosa
4          Iris-setosa
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: species, Length: 150, dtype: object
```

Visualizing data using Matplotlib

- Create a new file named week3_data_visualization in the same folder Week3Tutorial
- Read the iris dataset like in previous task

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

#read iris data
iris_dataset = pd.read_csv("iris.data", sep=',', names=["sepal_length", "sepal_width", "petal_length",
                                                        "petal_width", "species"])
```

- Pie char

```
iplot = iris_dataset['species']\
    .value_counts()\
    .plot(kind='pie',autopct='%.2f',figsize=(8, 8))

iplot.set_ylabel('')
```

- Boxplot

```
iris_dataset.boxplot(by="species", figsize=(12, 6))
```

- Scatterplot

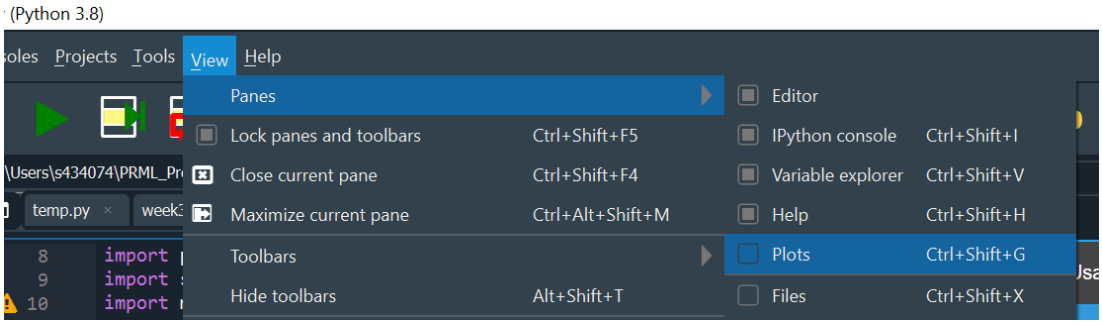
```
sns.set(style="darkgrid")
sc=iris_dataset[iris_dataset.species=='Iris-setosa'].plot(kind='scatter',x='sepal_length',y='sepal_width',
                                                          color='red',label='Setosa')
iris_dataset[iris_dataset.species=='Iris-versicolor'].plot(kind='scatter',x='sepal_length',y='sepal_width',
                                                           color='green',label='Versicolor',ax=sc)
iris_dataset[iris_dataset.species=='Iris-virginica'].plot(kind='scatter',x='sepal_length',y='sepal_width',
                                                          color='orange', label='virginica', ax=sc)
sc.set_xlabel('Sepal Length in cm')
sc.set_ylabel('Sepal Width in cm')
sc.set_title('Sepal Length Vs Sepal Width')
```

If you see this message:

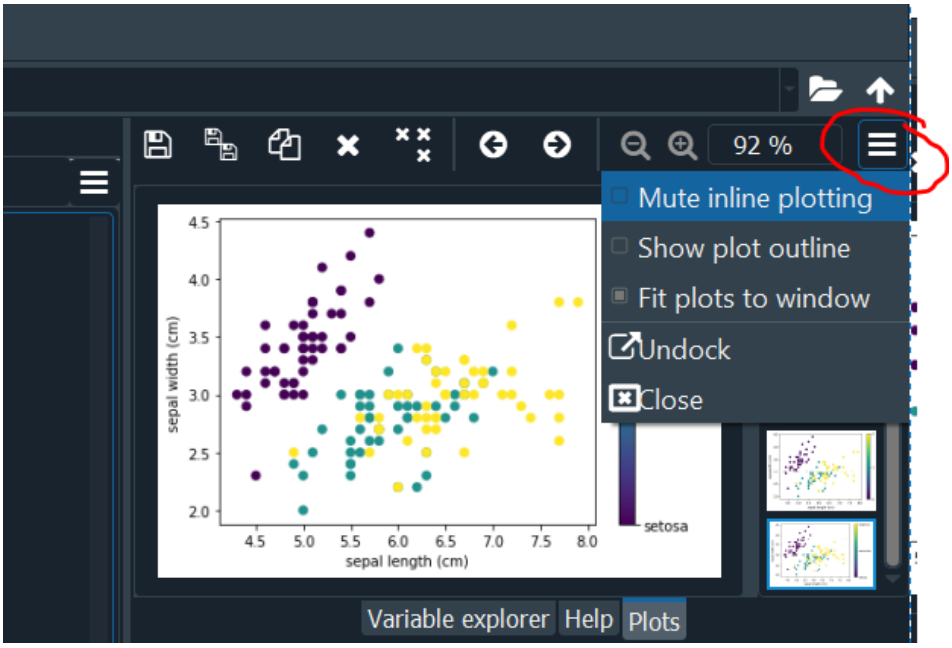
```
Figures now render in the Plots pane by default. To make them also
appear inline in the Console, uncheck "Mute Inline Plotting" under
the Plots pane options menu.
```

Please do these two steps below:

- Click on View and select Panes -> Plots



- Click on a dialog in a red circle and choose Mute inline plotting



Practise on a linear regression on multi-variable dataset

The hypothesis is that petal width depends on sepal length, sepal width and petal length

- Create a python file named LR_Example in the same Week3Tutorial folder
- Import library

```
import pandas as pd
import numpy as np

#metrics to evaluate the linear regression
from sklearn.metrics import mean_squared_error, mean_absolute_error

#split the data to train/test dataset
from sklearn.model_selection import train_test_split

#import the linear regression
from sklearn.linear_model import LinearRegression
```

- Read the iris data

```
#read iris data
iris_dataset = pd.read_csv("iris.data", sep=',', names=["sepal_length", "sepal_width", "petal_length",
                                                        "petal_width", "species"])
```

- Prepare data for training

```
#remove petal width from data
X=iris_dataset.drop(labels='petal_width', axis=1)

#remove species from data
X=X.drop(labels='species', axis=1)

#petal width values
y=iris_dataset['petal_width']

#split data into train/test dataset with ratio 3/1
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.25, random_state= 1)
```

Check the size of training and testing dataset

```
In [142]: X_train.shape
Out[142]: (112, 3)

In [143]: X_test.shape
Out[143]: (38, 3)
```

- Training with linear regression model

```
lre = LinearRegression()

#train the train data
lre.fit(X_train, y_train)
```

- Make a prediction

```
#predict on the test data
pred = lre.predict(X_test)
```

- Evaluate the model

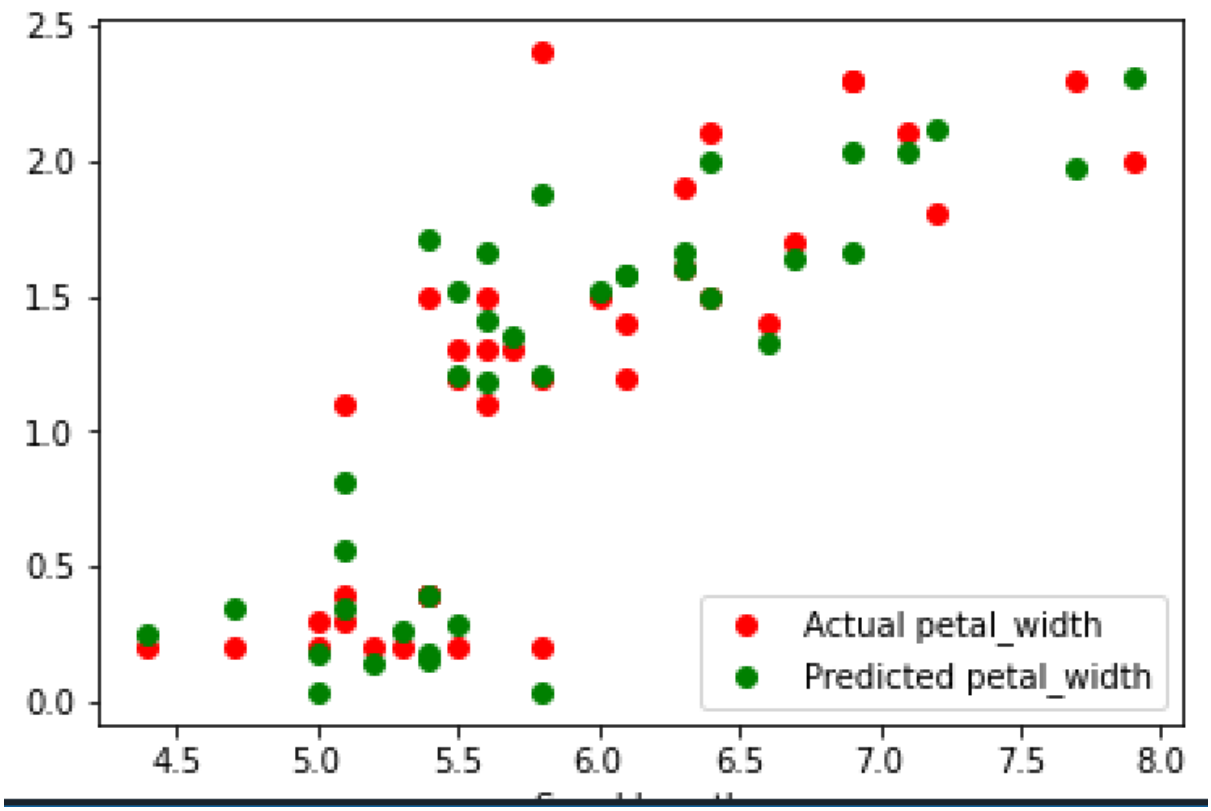
```
#Evaluate the prediction
print('Mean Absolute Error:', mean_absolute_error(y_test, pred))
print('Mean Squared Error:', mean_squared_error(y_test, pred))
print('Mean Root Squared Error:', np.sqrt(mean_squared_error(y_test, pred)))
abs(y_test-pred)
```

Output:

```
In [141]: runfile('C:/Users/s434074/PRML_Programming/Week2Tutorial/LR_Example.py', wdir='C:/Users/s434074/
PRML_Programming/Week2Tutorial')
Mean Absolute Error: 0.16984559986869163
Mean Squared Error: 0.048916591588477845
Mean Root Squared Error: 0.22117095557165242
```

- Visualization the prediction

```
plt.scatter(X_test[["sepal_length"]], y_test, color = "red",
            label = "Actual petal_width")
plt.scatter(X_test[["sepal_length"]], pred, color = "green",
            label = "Predicted petal_width")
plt.legend()
plt.xlabel('Sepal Length')
```



Practice on Linear Regression: Wine Quality dataset (Optional)
https://archive.ics.uci.edu/ml/datasets/wine+quality

Predict a wine quality based on 11 features:

1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 – alcohol

The hypothesis is that wine quality depends on all 11 features.

- Create a python file named wine_quality in the same Week3Tutorial folder
- Import library

```
import pandas as pd
import numpy as np

from sklearn.metrics import mean_squared_error, mean_absolute_error

#split the data to train/test dataset
from sklearn.model_selection import train_test_split

#import the linear regression
from sklearn.linear_model import LinearRegression
```

- Read the wine quality data

```
#read the wine quality data
wine_red_dataset = pd.read_csv("winequality-red.csv", sep=';')
wine_white_dataset = pd.read_csv("winequality-white.csv", sep=';')

wine_dataset = pd.concat([wine_red_dataset, wine_red_dataset])
```

- Inspect the data and 11 features by yourself
- Prepare data for training

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size= 0.25,
                                                    random_state= 1)
```

- Training with linear regression model

```
lre = LinearRegression()

#train the train data
lre.fit(X_train, y_train)
```

- Make a prediction

```
#predict on the test data
pred = lre.predict(X_test)
```

- Evaluate the model

```
#Evaluate the prediction
print('Mean Absolute Error:', mean_absolute_error(y_test, pred))
print('Mean Squared Error:', mean_squared_error(y_test, pred))
print('Mean Root Squared Error:', np.sqrt(mean_squared_error(y_test, pred)))
abs(y_test-pred)
```

Output:

```
Mean Absolute Error: 0.5074554493866952
Mean Squared Error: 0.426552687670651 96
Mean Root Squared Error: 0.65311001192039
```

- Print actual wine quality and predicted one

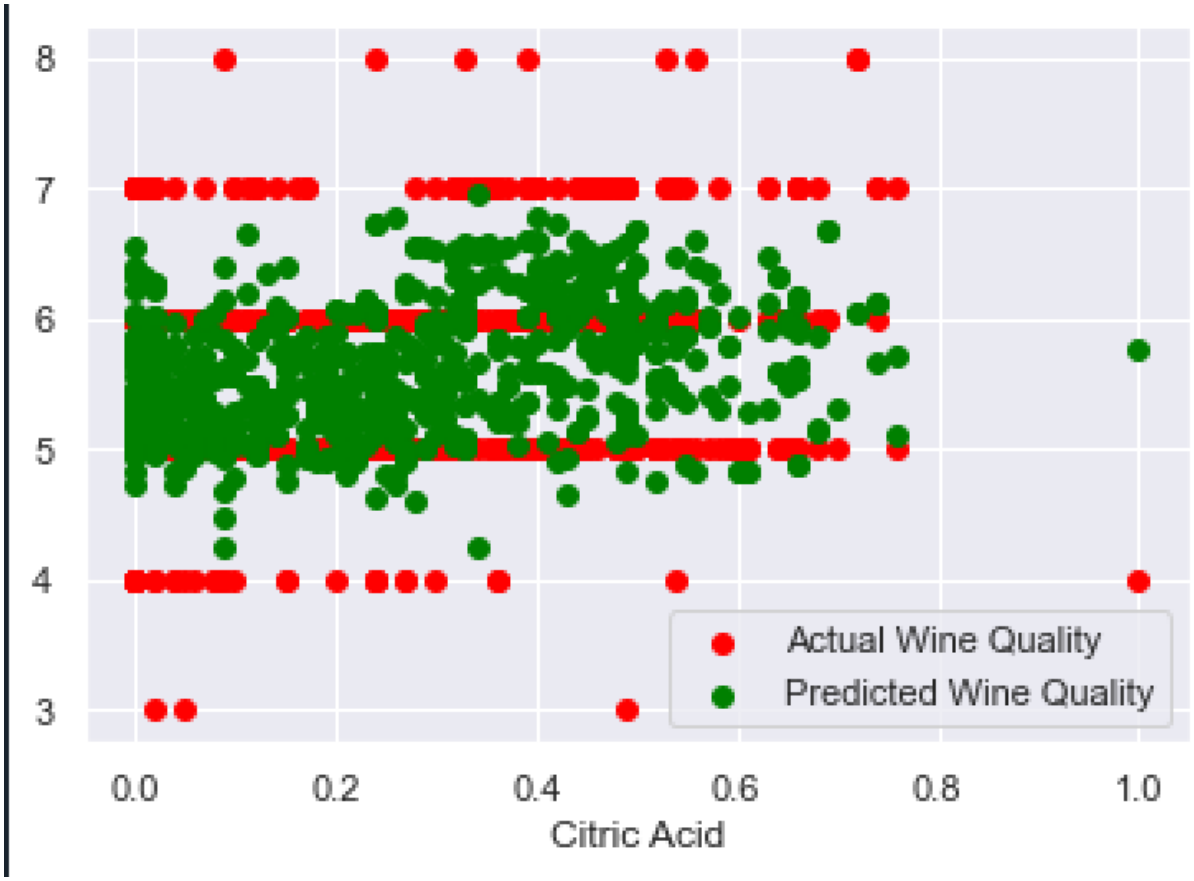
```
print(pd.DataFrame({'Actual': y_test, 'Predicted': pred}))
```

- Print weight and coefficients

```
print(lre.intercept_)
print(lre.coef_)
```

- Visualize the output for the feature “Citric Acid”

```
plt.scatter(X_test[['citric acid']], y_test, color = "red",
            label = "Actual Wine Quality")
plt.scatter(X_test[['citric acid']], pred, color = "green",
            label = "Predicted Wine Quality")
plt.legend()
plt.xlabel('Volatile Acidity')
```



Question:

1. Investigate an effect of testing size to model's performance
2. Build a model to train and make a prediction for your selected dataset.
