part A) Exploratory Data Analysis:

Exercise 1:

use the data set titanic from seaborn library via following code:

```
import seaborn as sns
titanic = sns.load_dataset('titanic')
```

perform the following:

1. Select your preferred target feature for the goal of this activity.
2. Create both the features matrix and the target matrix. Make sure that you store the data from the features matrix in a variable, **X**, and the data from the target matrix in another variable, **Y**.
3. Print out the shape of each of the matrices, which should match the following values:
4. Check for missing values and outliers in all the features of the features matrix (X). Choose a methodology to handle them.
5. Convert all text features into its numeric representation.
6. Rescale your data, either by normalizing or standardizing.

Exercise 2:

use the data set named "house_prices.csv" in the accompanying folder (also available at https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data or https://github.com/shobhit-nigam/alstom/tree/master/DS_june/exercises)

Perform the following:

1. Read the data.
2. Use pandas' **.info()** and **.describe()** methods to view the summary statistics of the dataset.
3. Find the total count and total percentage of missing values in each column of the DataFrame and display them for columns having at least one null value, in descending order of missing percentages.
4. Plot the nullity matrix and nullity correlation heatmap.
5. Delete the columns having more than 80% of values missing.
6. Replace null values in the **FireplaceQu** column with **NA** values.

Exercise 3:

Use the data set named "house_prices.csv" in the accompanying folder (also available at https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data or https://github.com/shobhit-nigam/alstom/tree/master/DS_june/exercises)

Perform the following:

1. Plot a histogram using Matplotlib for the target variable, **SalePrice**.
2. Find the number of unique values within each column having an object type.
3. Create a DataFrame representing the number of occurrences for each categorical value in the **HouseStyle** column.
4. Plot a pie chart representing these counts.
5. Find the number of unique values within each column having a number type.
6. Plot a histogram using Seaborn for the **LotArea** variable.

7. Calculate the skew and kurtosis values for the values in each column.

Exercise 4:

Use the data set named “house_prices.csv” in the accompanying folder (also available at https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data or https://github.com/shobhit-nigam/alstom/tree/master/DS_june/exercises)

1. Plot the correlation heatmap for the dataset.
2. Plot a more compact heatmap having annotations for correlation values using the following subset of features:
   Copy

```
feature_subset = [
    'GarageArea',
'GarageCars','GarageCond','GarageFinish','GarageQual','GarageType',
    'GarageYrBlt','GrLivArea','LotArea','MasVnrArea','SalePrice'
]
```

3. Display the pairplot for the same subset of features, with the KDE plot on the diagonals and scatter plot elsewhere.
4. Create a boxplot to show the variation in **SalePrice** for each category of **GarageCars**.
5. Plot a line graph using Seaborn to show the variation in **SalePrice** for older and more recently built flats.

part B) Regression Analysis:

Exercise 5:

Use the data set named "austin_weather.csv" in the accompanying folder (also available at https://github.com/shobhit-nigam/alstom/tree/master/DS_june/exercises)

1. Load the dataset into a pandas DataFrame from the CSV file.
2. We only need the **Date** and **TempAvgF** columns; remove all others from the dataset.
3. Initially, we will only be interested in the first year's data, so we need to extract that information only. Create a column in the DataFrame for the year value and extract the year value as an integer from the strings in the **Date** column and assign these values to the **Year** column.
   Note that temperatures are recorded daily.
4. Repeat this process to extract the month values and store the values as integers in a **Month** column.
5. Copy the first year's worth of data to a DataFrame.
6. Compute a 20-day moving average filter.
7. Plot the raw data and moving average signal, with the **x** axis being the day number in the year.

Note: create two csv with modified data: "measurements.csv" and "rolling.csv" which will be used for the next exercise.

Exercise 6:

Use the data set named "austin_weather.csv" in the accompanying folder (also available at https://github.com/shobhit-nigam/alstom/tree/master/DS_june/exercises).
Finish Exercise 5 and store the two csv with modified data: "measurements.csv" and "rolling.csv", and use them for this exercise. If you have skipped Exercise 4, then download the two CSVs from https://github.com/shobhit-nigam/alstom/tree/master/DS_june/exercises or from accompanying folder.

Run the following code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression


# Loading the data from activity 5
df = pd.read_csv('measurements.csv')
df_first_year = df[:365]
rolling = pd.read_csv('rolling.csv')
window = 20
```

Perform the following:

1. Visualize the measurements.
2. Visualize the rolling average values.
3. Create a linear regression model using the default parameters, that is, calculate a **y** intercept for the model and do not normalize the data.
4. Now fit the model, where the input data is the day number for the year (1 to 365) and the output is the average temperature. To make later calculations easier, insert a column (**DayOfYear**) that corresponds with the day of the year for that measurement.

5. Fit the model with the **DayOfYear** values as the input and **df_first_year.TempAvgF** as the output.
6. Print the parameters of the model.
7. Let's check the trendline provided by the model. Plot this simply using the first, middle, and last values (days in years) in the linear equation.
8. Plot the values with the trendline.
9. Evaluate the performance of the model.
10. Let's check how well the model fits the data. Calculate the r2 score to find out.

Exercise 7:

Use the data set named "austin_weather.csv" in the accompanying folder (also available at https://github.com/shobhit-nigam/alstom/tree/master/DS_june/exercises).
Finish Exercise 5 & 6 and store the csv with modified data: "measurements.csv", "rolling.csv" and "first_year.csv" , and use them for this exercise.
If you have skipped Exercise 5 & 6, then download the three CSVs from https://github.com/shobhit-nigam/alstom/tree/master/DS_june/exercises or from accompanying folder.
Run the following code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Loading the data from activity 5
df = pd.read_csv('measurements.csv')
df_first_year = pd.read_csv('first_year.csv')
rolling = pd.read_csv('rolling.csv')
window = 20

# Trendline values
trend_x = np.array([
    1,
    182.5,
    365
])
```

perform the following:

1. Plot the raw data (**df**) and moving average (**rolling**).
2. Looking at the result of the previous step, there seems to be an inflection point around day 250. Create a dummy variable to introduce this feature into the linear model.
3. Check the first and last samples to confirm that the dummy variable is correct.
4. Use a least squares linear regression model and fit the model to the **DayOfYear** values and the dummy variable to predict **TempAvgF**.
5. Compute the R2 score.
6. Using the **DayOfYear** values, create a set of predictions using the model to construct a trendline.
7. Plot the trendline against the data and moving average.

Exercise 8:

Use the data set named “austin_weather.csv” in the accompanying folder (also available at https://github.com/shobhit-nigam/alstom/tree/master/DS_june/exercises).
Finish Exercise 5 & 6 and store the csv with modified data: “measurements.csv”, “rolling.csv” and “first_year.csv” , and use them for this exercise.
If you have skipped Exercise 5 & 6, then download the three CSVs from https://github.com/shobhit-nigam/alstom/tree/master/DS_june/exercises or from accompanying folder.
Run the following code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression


# Loading the data from activity 5
df = pd.read_csv('measurements.csv')
df_first_year = pd.read_csv('first_year.csv')
rolling = pd.read_csv('rolling.csv')
window = 20

# Trendline values
trend_x = np.array([
    1,
    182.5,
    365
])
```

Perform the following:

1. Use a sine curve function as the basis of the model.
2. Print the parameters of the model.
3. Compute the r2 value to measure the performance.
4. Construct the trendline values.
5. Plot the trendline with the raw data and the moving average.

part C) Classification:

Run the following code for Exercise 9, 10, 11, 12:

```
import struct
import numpy as np
import gzip
import urllib.request
import matplotlib.pyplot as plt
from array import array
from sklearn.linear_model import LogisticRegression
```

```
request = urllib.request.urlopen('http://yann.lecun.com/exdb/mnist/train-images-
idx3-ubyte.gz')

with open('train-images-idx3-ubyte.gz', 'wb') as f:
    f.write(request.read())

request = urllib.request.urlopen('http://yann.lecun.com/exdb/mnist/t10k-images-
idx3-ubyte.gz')

with open('t10k-images-idx3-ubyte.gz', 'wb') as f:
    f.write(request.read())
request = urllib.request.urlopen('http://yann.lecun.com/exdb/mnist/train-labels-
idx1-ubyte.gz')

with open('train-labels-idx1-ubyte.gz', 'wb') as f:
    f.write(request.read())

request = urllib.request.urlopen('http://yann.lecun.com/exdb/mnist/t10k-labels-
idx1-ubyte.gz')

with open('t10k-labels-idx1-ubyte.gz', 'wb') as f:
    f.write(request.read())
```

```
with gzip.open('train-images-idx3-ubyte.gz', 'rb') as f:
    magic, size, rows, cols = struct.unpack(">IIII", f.read(16))
    img = np.array(array("B", f.read())).reshape((size, rows, cols))

with gzip.open('train-labels-idx1-ubyte.gz', 'rb') as f:
    magic, size = struct.unpack(">II", f.read(8))
    labels = np.array(array("B", f.read()))

with gzip.open('t10k-images-idx3-ubyte.gz', 'rb') as f:
    magic, size, rows, cols = struct.unpack(">IIII", f.read(16))

    img_test = np.array(array("B", f.read())).reshape((size, rows, cols))

with gzip.open('t10k-labels-idx1-ubyte.gz', 'rb') as f:
    magic, size = struct.unpack(">II", f.read(8))
    labels_test = np.array(array("B", f.read()))
```

Exercise 9:

After having executed the code mentioned at the start of part C, perform the following:

1. Visualize a sample of the data.
2. Construct a linear classifier model to classify the digits zero and one. The model we are going to create is to determine whether the samples are either the digits zero or one. To do this, we first need to select only those samples.
3. Visualize the selected information with images of one sample of zero and one sample of one.
4. In order to provide the image information to the model, we must first flatten the data out so that each image is 1 x 784 pixels in shape.
5. Let's construct the model; use the **LinearRegression** API and call the **fit** function.
6. Determine the R2 score against the training set.
7. Determine the label predictions for each of the training samples, using a threshold of 0.5. Values greater than 0.5 classify as one; values less than or equal to 0.5 classify as zero.
8. Compute the classification accuracy of the predicted training values versus the ground truth.
9. Compare the performance against the test set.

Exercise 10:

After having executed the code mentioned at the start of part C, perform the following:

1. Plot a number of different features versus the allocated species classifications, for example, sepal length versus petal length and species. Visually inspect the plots and look for any patterns that could indicate separation between each of the species.
2. Select the features by writing the column names in the following list:
   Copy

```
selected_features = [
    '', # List features here
]
```

3. Before we can construct the model, we must first convert the **species** values into labels that can be used within the model. Replace the **Iris-setosa** species string with the value **0**, the **Iris-versicolor** species string with the value **1**, and the **Iris-virginica** species string with the value **2**.
4. Create the model using the **selected_features** and the assigned **species** labels.
5. Compute the accuracy of the model against the training set.
6. Construct another model using your second choice **selected_features** and compare the performance.
7. Construct another model using all available information and compare the performance.

Exercise 11:

After having executed the code mentioned at the start of part C, perform the following:

1. Visualize a sample of the data.
2. Construct a K-NN classifier, with three nearest neighbors to classify the MNIST dataset. Again, to save processing power, randomly sample 5,000 images for use in training.
3. In order to provide the image information to the model, we must first flatten the data out so that each image is 1 x 784 pixels in shape.
4. Build the three-neighbor KNN model and fit the data to the model. Note that, in this activity, we are providing 784 features or dimensions to the model, not simply 2.
5. Determine the score against the training set.
6. Display the first two predictions for the model against the training data.
7. Compare the performance against the test set.

Exercise 12:

After having executed the code mentioned at the start of part C, perform the following:

1. Have two different datasets for data & target (X & Y).
2. Import the **train_test_split** function from scikit-learn's **model_selection** package:
   Copy

```
from sklearn.model_selection import train_test_split
```

3. Perform a first split of the data using the function that we just imported, with test size as 0.2

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2)
```

4. Print the shape of all the four matrices.
5. create a validation set (dev set), by splitting the training data further (use the **train_test_split** function to divide the train sets)

```
X_train, X_dev, Y_train, Y_dev = train_test_split(X_train, Y_train,
test_size = 0.25)
```

6. Use any of the data sets provided in th accompanying folder and split the data into training, validation & testing set. Print the matrices.

part D)

Exercise 13:

Use the data set named "wholesale customers data.csv" in the accompanying folder (also available at http://archive.ics.uci.edu/ml/machine-learning-databases/00292/ or https://github.com/shobhit-nigam/alstom/tree/master/DS_june/exercises).
Perform the following:

1. Calculate the average distance of the data points from its centroid in relation to the number of clusters. Based on this distance, select the appropriate number of clusters to train the model.
2. Train the model using KMeans from sklearn and assign a cluster to each data point in your dataset. Plot the results.

```
from sklearn.cluster import KMeans
```

3. Train the model using MeanShift from sklearn and assign a cluster to each data point in your dataset. Plot the results.

```
from sklearn.cluster import MeanShift
```

Exercise 14:

Use the data set named "adult_data.csv" in the accompanying folder (also available at https://github.com/shobhit-nigam/alstom/tree/master/DS_june/exercises).
Perform the following:

1. preprocess the Census Income Dataset, separate the features from the target by creating the variables **X** and **Y**.
2. Divide the dataset into training, validation, and testing sets, using a split ratio of 10%.
3. Import the Gaussian Naïve Bayes class, and then use the **fit** method to train the model over the training sets (**X_train** and **Y_train**).
4. Finally, perform a prediction using the model that you trained previously, for a new instance with the following values for each feature: 39, 6, 13, 4, 0, 2174, 0, 40, 38. The prediction for the individual should be equal to zero, meaning that the individual most likely has an income less than or equal to 50K.

Exercise 15:

Use the data set named "fertility_Diagnosis.csv" in the accompanying folder (also available at https://github.com/shobhit-nigam/alstom/tree/master/DS_june/exercises).
Perform the following:

1. Import **pandas** and read the **fertility_Diagnosis** dataset. Make sure to add the argument **header** equal to **None** to the **read_csv** function, considering that the dataset does not contain a header row:

```
import pandas as pd
data = pd.read_csv("fertility_Diagnosis.csv", header=None)
```

2. Split the data into **X** and **Y**, considering that the class label is found under the column with index equal to 9.
3. Import scikit-learn's **DecisionTreeClassifier** class. Then, initialize it and use the **fit** function to train the model using **X** and **Y**:
4. Perform a prediction by using the model that you trained, for the values 0.33, 0.69, 0, 1, 1, 0, 0.8, 0, 0.88.
5. Go back to step 2 and re start step 3 by Importing scikit-learn's **SVC** class. Then, initialize it and use the **fit** function to train the model using **X** and **Y**.

6. Perform a prediction by using the model that you trained, for the values 0.33, 0.69, 0, 1, 1, 0, 0.8, 0, 0.88.

The author has used datasets, exercises charts, diagrams, content etc from his own exercises build during his sessions and also has a mixture of some material taken from the internet (freely available) from resources like:

1. kaggle.com
2. medium.com
3. towardsdatascience.com
4. archive.ics.uci.edu/ml/index.php
5. official documentations of various open source libraries, inclusive of (but not restricted to):
    a. matplotlib.org
    b. scipy.org
    c. numpy.org
    d. pydata.org (pandas & seaborn)
6. Slide Share
7. Purchasable Books like:
    a. Data Science from Scratch: First Principles with Python by Joel Grus
    b. Storytelling with Data: A Data Visualization Guide for Business Professionals by Cole Nussbaumer Knaflic
    c. Applied Supervised Learning with Python by Benjamin Johnston
    d. Data Science beginners book by Andrew Park and Russell Newton
    e. Machine Learning Fundamentals by Hyatt Saleh
    f. The Data Science Workshop by Anthony So, Thomas V. Joseph, Robert Thas John, Andrew Worsley, Dr. Samuel Asare
8. various resource feeds from Data scientists and their blogs.