Exploratory Data Analysis

Important Libraries

1. Pandas

Pandas is a Python programming language for **data manipulation** and **data analysis**. In particular, it offers data structures and operations for manipulating numerical tables and time series.

2. Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays

3. Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms

4. Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline #%matplotlib inline will lead to static images of your plot embedded in t
he notebook
```

```
UsageError: unrecognized arguments: #%matplotlib inline will lead to static images of your pl
ot embedded in the notebook
```

Objective

Our Objectives are as follows:-

Data cleansing

Data cleansing or **data cleaning** is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

Data wrangling

Data wrangling, sometimes referred to as data munging, is the process of transforming and mapping data from one "raw" data form into another format with the intent of making it more appropriate and valuable for a variety of downstream purposes such as analytics.This may include further **munging**, **data visualization**, data aggregation, training a statistical model, as well as many other potential uses.

Metadata

Our Metadata contains 10015 unique image_id but 7470 unique lesion_id. Most of the cancer images are from 'nv' cancer type which is around 67% , 'mel' cancer images are around 11% and other 5 cancer type (bkl,dcl,df,akiec, vasc) are 22% as whole.

In [2]:
```
data=pd.read_csv("C:/Users/imfal/skincancer/image classification project/HAM10000_metadata.c
sv")
data.sort_values("lesion_id", inplace = True)
```

In [3]:
```
data.head() # original data has matrix 10015x07
```

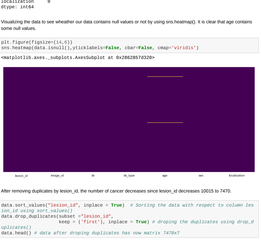
Out[3]:

	lesion_id	image_id	dx	dx_type	age	sex	localization
9197	HAM_0000000	ISIC_0025498	nv	histo	80.0	male	back
9198	HAM_0000000	ISIC_0025340	nv	histo	80.0	male	back
726	HAM_0000001	ISIC_0027959	bkl	histo	70.0	female	face
1661	HAM_0000002	ISIC_0032622	mel	histo	65.0	female	lower extremity
1660	HAM_0000002	ISIC_0033840	mel	histo	65.0	female	lower extremity

By Visualizing data with respect to cancer type dx, which are 7 in numbers. We can clearly see that **nv** group has (around 67%) infected more people in our data.

In [4]:
```
# Looking for which cancer has infected people more.
plt.figure(figsize=(14,6))
sns.set_style('whitegrid')
sns.countplot(x='dx',data=data)
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2826df55e48>
```



isnull()

While making a Data Frame from a csv file, many blank columns are imported as null value into the Data Frame which later creates problems while operating that data frame. Pandas isnull() and notnull() methods are used to check and manage NULL values in a data frame.

In [5]:
```
data.isnull().head() # looking for null values
```

Out[5]:

	lesion_id	image_id	dx	dx_type	age	sex	localization
9197	False	False	False	False	False	False	False
9198	False	False	False	False	False	False	False
726	False	False	False	False	False	False	False
1661	False	False	False	False	False	False	False
1660	False	False	False	False	False	False	False

In [6]:
```
data.isnull().sum() # counting null values column-wise using sum() library with isnull().
                    # And there are 57 null values in column ""age""
```

Out[6]:
```
lesion_id      0
image_id       0
dx             0
dx_type        0
age           57
sex            0
localization   0
dtype: int64
```

Visualizing the data to see wheather our data contains null values or not by using sns.heatmap(). It is clear that age contains some null values.

In [7]:
```
plt.figure(figsize=(14,6))
sns.heatmap(data.isnull(),yticklabels=False, cbar=False, cmap='viridis')
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28628575320>
```


After removing duplicates by lesion_id, the number of cancer decreases since lesion_id decreases 10015 to 7470.

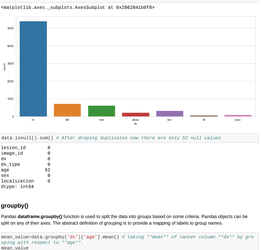
In [8]:
```
data.sort_values("lesion_id", inplace = True) # Sorting the data with respect to column les
ion_id using sort_values()
data.drop_duplicates(subset ='lesion_id',
                     keep = 'first', inplace = True) # dropping the duplicates using drop_d
uplicates()
data.head() # data after droping duplicates has now matrix 7470x7
```

Out[8]:

	lesion_id	image_id	dx	dx_type	age	sex	localization
9197	HAM_0000000	ISIC_0025498	nv	histo	80.0	male	back
726	HAM_0000001	ISIC_0027959	bkl	histo	70.0	female	face
1661	HAM_0000002	ISIC_0032622	mel	histo	65.0	female	lower extremity
3374	HAM_0000003	ISIC_0027880	nv	follow_up	55.0	male	trunk
4918	HAM_0000004	ISIC_0024640	nv	follow_up	40.0	female	back

In [9]:
```
plt.figure(figsize=(14,6))
sns.set_style('whitegrid')
sns.countplot(x='dx',data=data)
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2862841b0f0>
```



In [10]:
```
data.isnull().sum() # After droping duplicates now there are only 52 null values
```

Out[10]:
```
lesion_id      0
image_id       0
dx             0
dx_type        0
age           52
sex            0
localization   0
dtype: int64
```

groupby()

Pandas **dataframe.groupby()** function is used to split the data into groups based on some criteria. Pandas objects can be split on any of their axes. The abstract definition of grouping is to provide a mapping of labels to group names.

In [11]:
```
mean_value=data.groupby('dx')['age'].mean() # taking ""mean"" of cancer column ""dx"" by gro
uping with respect to ""age"".
mean_value
```

Out[11]:
```
dx
akiec    60.557018
bcc      66.896624
bkl      64.384158
df       55.000000
mel      61.299389
nv       47.257974
vasc     52.500000
Name: age, dtype: float64
```

In [12]:
```
median_value=data.groupby('dx')['age'].median()
median_value  # taking ""median"" of cancer column ""dx"" by grouping with respect to ""age
```

Out[12]:
```
dx
akiec    67.5
bcc      70.0
bkl      65.0
df       55.0
mel      65.0
nv       45.0
vasc     45.0
Name: age, dtype: float64
```

Filling Null Values

Here it should be notice that filling of null values is taking place in column age with respect to cancer type dx. Meaning, the median of age of akiec group cancer patient will only fill null values of akiec age group and so other.

In [13]:
```
# Filling the null values by using median_value
data["age"] = data.groupby("dx").transform(lambda x: x.fillna(x.median()))
data.head()
```

Out[13]:

	lesion_id	image_id	dx	dx_type	age	sex	localization
9197	HAM_0000000	ISIC_0025498	nv	histo	80.0	male	back
726	HAM_0000001	ISIC_0027959	bkl	histo	70.0	female	face
1661	HAM_0000002	ISIC_0032622	mel	histo	65.0	female	lower extremity
3374	HAM_0000003	ISIC_0027880	nv	follow_up	55.0	male	trunk
4918	HAM_0000004	ISIC_0024640	nv	follow_up	40.0	female	back

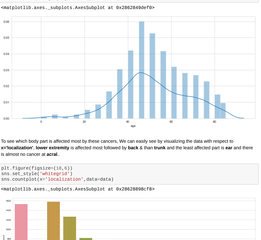
In [14]:
```
data.isnull().sum() # now we can clearly see that there is no null values remaining.
```

Out[14]:
```
lesion_id      0
image_id       0
dx             0
dx_type        0
age            0
sex            0
localization   0
dtype: int64
```

Here,by visualizing distribution plot of age, We can clearly see that the highest number of cancer patient are from the age group (40,60) and highest number of cancer patient are of age 45.

In [15]:
```
plt.figure(figsize=(14,6))
sns.distplot(data['age'],bins=range(0,100,5),kde=True)
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2862849def0>
```



To see which body part is affected most by these cancers, We can easily see by visualizing the data with respect to **x='localization'**, **lower extremity** is affected most followed by **back** & than **trunk** and the least affected part is **ear** and there is almost no cancer as **acral**.

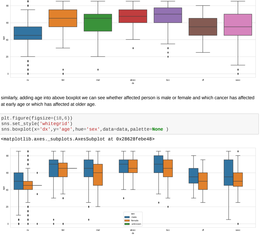
In [16]:
```
plt.figure(figsize=(10,6))
sns.set_style('whitegrid')
sns.countplot(x='localization',data=data)
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28628898cf0>
```

Now, Visualizing the data with respect to **x='localization'** and cancers type **dx**. We can clearly see at **lower extremity nv** cancer has affected most followed by **bkl** & **mel**. Similarly at **back** side **nv** has affected most followed by **mel**.**bkl**. But when we look at **ear** , there we can see this is least affected part of our body and mainly affected by **nv** & **mel** and there is almost no cancer at **acral**.

In [17]:
```
plt.figure(figsize=(10,6))
sns.set_style('whitegrid')
sns.countplot(x='localization',hue='dx',data=data,palette='Reds' )
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2862846abcc0>
```



Now by plotting boxplot with respect to cancers type **x='dx'** with **age**. we can see **nv** has affected people at early age around (40,55), while **vasc** has large age interval of affecting people of age around (45,70) and **bcc** has affected mostly older age people around (60, 77).

In [18]:
```
plt.figure(figsize=(10,6))
sns.set_style('whitegrid')
sns.boxplot(x='dx',y='age',data=data,palette='Reds' )
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28628f0c438>
```


similarly, adding age into above boxplot we can see whether affected person is male or female and which cancer has affected at early age or which has affected at older age.

In [19]:
```
plt.figure(figsize=(10,6))
sns.set_style('whitegrid')
sns.boxplot(x='dx',y='age',hue='sex',data=data,palette='Reds' )
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28628feb4e0>
```