1. Introduction

Our group work on the image classification project which identifies artist from art based on neural networks by using the magic of deep Learning. The datasets we use gathered a collection of artworks of the 50 most influential artists of all time. The process includes exploratory data analysis, preprocessing, neural network model training and performance measurement.

The dataset we use contains three files: one csv file contains all artists' information and two zip files contain 50 folders, one for each artist. By breaking the column of years, we would be able to create another 3 variables: birth, death and age. Based on the lifetime of each artist, a new variable called age_group is created for further analysis.

2. Description of individual work

- Collecting dataset and google for artists back group for further understanding
- Analyzing dataset in different aspects like age, nationality and genre.
- Visualizing data by different methods like boxplot, barplot, voilinplot, displot and so on to get a better understanding dataset.
- Using Keras MLP model from exam one for image classification.
- Working closely with other team member to write final report and prepare presentation slides.
- Working as a good team member by setting up report template, scheduling team meeting, making reservation and so on.

3. Description of portion

Since there is one csv file, I could do the simple EDA on local PC. When it comes to modeling, then move it to GCP.

- By breaking the column of years, we would be able to create another 3 variables: birth, death and age.

```
df1_year = pd.DataFrame(df1.years.str.split(' ',2).tolist(),columns = ['birth','-','death'])
df1_year.drop(["-"], axis=1,inplace=True)
df1["birth"] = df1_year.birth
df1["death"] = df1_year.death
df1["age"] = df1["death"].apply(lambda x: int(x)) - df1["birth"].apply(lambda x: int(x))
```

- Based on the lifetime of each artist, a new variable called age_group is created for further analysis. Instead of using minimum, 25%, 50%, 75% and maximum as the break points, I will the social general method to group age.

```
bins=[18,40,65,80,98]
group = ["Teenage","Young Adult","Adult","elderly"]
df1['age_structure'] = pd.cut(df1['age'], bins, labels = group)
```

- From distribution painting, we could see some outliers from the long tail

```
# Painting distribution
plt.figure(figsize=(16,10))
sns.violinplot(df1['paintings'], color = 'purple')
plt.xlabel('paintings')
plt.ylabel('Frequency')
plt.title('Paintings distribution')
plt.show()
```

- Order the painting number in decedent order, I could see most of the artist had painting less than 200. As a result, we could pick the top 10 artists with the most painting for model training

```
# Pick up top 10 to create a new data frame and use weight to salve unbalance data set
top10 = df1.sort_values(by=["paintings"], ascending=False).head(10).reset_index()
top10.iloc[4, 1] = "Albrecht_Dürer".replace("_", " ")
top10.iloc[3, 1] = "Pierre-Auguste Renoir".replace("_", " ")

top10_name=['Vincent_van_Gogh', 'Edgar_Degas' ,'Pablo_Picasso', 'Pierre-Auguste_Renoir','Albrecht_Dürer',
            'Paul_Gauguin', 'Francisco_Goya', 'Rembrandt', 'Alfred_Sisley', 'Titian']
```

- Print 6 paintings which are random picked from the 10 folders of the top 10 artists.

```
for j in range(5):
    j=0
    for i in range(6):
        artist = random.choice(top10_name)
        image = random.choice(os.listdir(os.path.join(pic_dir, artist)))
        image_file = os.path.join(pic_dir, artist, image)
        image = plt.imread(image_file)
        axes[i].imshow(image)
        axes[i].set_title(artist.replace('_', ' '))
        axes[i].axis('off')
    j += 1
    plt.show()
```

- A few artists had multiple nationalities. At first, I thought break it into multiple columns and do one-hot-encoding. Later I found out it is meaningless since multi-nationalities should be considered as one special situation.

```
# Nationality Analysis
df2 = pd.DataFrame(df1['nationality'].str.split(',',2).tolist(), columns = ['n1','n2','n3'])
df1 = pd.concat([df1, df2], axis=1)
print("All Nations", pd.unique(df1[['n1', 'n2', 'n3']].values.ravel('K')))
```

- Same thing as nationality, a couple artists were counted as multiple genre. I will treat the multi-genre as one special situation without breaking it.

```
# Genre Analysis
df3 = pd.DataFrame(df1['genre'].str.split(',',2).tolist(), columns = ['g1','g2','g3'])
df1 = pd.concat([df1, df3], axis=1)
print("All Genres", pd.unique(df1[['g1', 'g2', 'g3']].values.ravel('K')))
```

- The Train dataset has 2 types of file, csv and jpg. By using for loop and append functions, setting all the jpg files as variable x and csv file as variable y. Convert the y to an array which will be the target for our images x. Both x and y are NumPy arrays.
- By setting the RESIZE equals to (224,224), all jpg files could be set as arrays at size of 224*224.

```
resize = (224,224)
imgs = []
label = []
for name in top10_name:
    for path in [f for f in os.listdir("/home/ubuntu/Final Project/best-artworks-of-all-time/images/images/" + name)]:
        img = cv2.resize(cv2.imread("/home/ubuntu/Final Project/best-artworks-of-all-time/images/images/" + name + "/" + path), resize)
        imgs.append(img)
        label.append(name)
x = np.array(imgs)
y = np.array(label)
print("First checking:", x.shape, y.shape)
```

- One-hot-encoding method is used on multi-class variable y by function LabelEncoer and to_categorical(). First, I handle the multiclass variable y by first encoding the strings consistently to integers using the scikit-learn class LabelEncoder. Second, I will convert the vector of integers to a one hot encoding by the function to_categorical() in the later step.

```
le = LabelEncoder()
le.fit(['Vincent_van_Gogh', 'Edgar_Degas' ,'Pablo_Picasso', 'Pierre-Auguste_Renoir','Albrecht_Dürer', 'Paul_Gauguin', 'Francisco_Goya',
y = le.transform(y)
print(x.shape, y.shape)
```

- Splitting the train dataset into 2 datasets for the purpose of training and testing by using function train_test_spllit. The stratifying percentage is set to 0.2.

```
# %%  Data Prep ---------------------------
print("Double checking shapes", x.shape, y.shape)
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=SEED, test_size=0.2, stratify=y)
x_train, x_test = x_train.reshape(len(x_train), -1), x_test.reshape(len(x_test), -1)
x_train, x_test = x_train/255, x_test/255
y_train, y_test = to_categorical(y_train, num_classes=10), to_categorical(y_test, num_classes=10)
print("Double checking shapes", x_train.shape, y_train.shape, x_test.shape, y_test.shape)
```

- The learning rate is as small as 0.001 to avoid this model from converging too quickly to a suboptimal solution. The Epoch can be up to 100. The dropout rate is set to 0.2.

```
# %%  Hyper Parameters --------------------
LR = 1e-3
N_NEURONS = 400
N_EPOCHS = 100
BATCH_SIZE = 512
DROPOUT = 0.2
```

- Sequential model is used for building the network. In this Sequential model, layers are stacked up by adding the 2 desired layers one by one. By building a feedforward network, all the neurons from one layer are connected to the neurons in the previous layer by the Dense Layer. The activation function is the relu activation function to introduce non-linearity to the model. The relu activation function could help the network learn non-linear decision boundaries. The second layer uses a softmax layer as the output is a multiclass variable. Note that softmax activation function in the output layer could ensure the output values are in the range of 0 and 1.
- Next, the MLP network will uses the popular Adam gradient descent optimization algorithm with a logarithmic loss function called categorical_crossentropy. It could tell from the following code that I chose the Adam as the optimizer and specify the categorical cross entropy as loss type since it is a multiclass classification.
- After specifying the optimizer and loss type, this model is ready for training. I call the training process by using the Keras function fit (). In previous step, I set up the number of epochs to 100 since there is enough computer power set on Google cloud. As a result, the whole dataset will be fed to the network 100 times.

```
# %%  Model  ----------------
model = Sequential([
    Dense(N_NEURONS, input_dim = 7500, activation="relu"),
    Dense(10, activation="softmax")
])
model.compile(optimizer=Adam(lr=LR), loss="categorical_crossentropy", metrics=["accuracy"])

model.fit(x_train, y_train, batch_size=BATCH_SIZE, epochs=N_EPOCHS, validation_data=(x_test, y_test),
          callbacks=[ModelCheckpoint("finalproject.hdf5", monitor="val_loss", save_best_only=True)])
```

- Checking the performance on the test dataset using the evaluate () method and printing out the final accuracy, Cohen Kappa Score and F1 score.

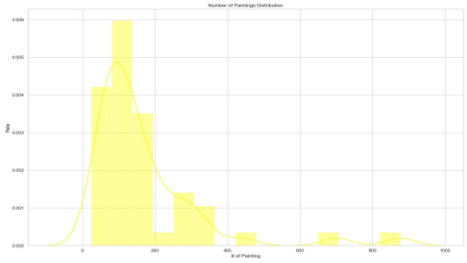
```
print("Final accuracy on validations set:", 100*model.evaluate(x_test, y_test)[1], "%")
print("Cohen Kappa", cohen_kappa_score(np.argmax(model.predict(x_test),axis=1),np.argmax(y_test,axis=1)))
print("F1 score", f1_score(np.argmax(model.predict(x_test),axis=1),np.argmax(y_test, axis=1), average = 'macro'))
```

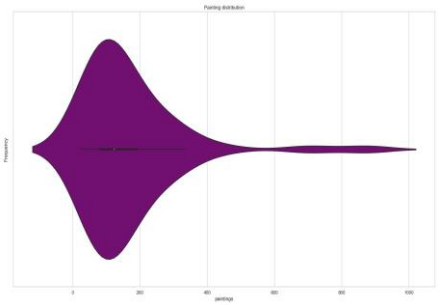
- The following is the prediction function which is designed to predict the output y by using the trained model in last step. From the path, images are collected and saved in x as input variable. The input x is updated as NumPy array and set into model to predict output y.

```
def predict(paths):
    x = []
    for path in paths:
        x.append(cv2.resize(cv2.imread(path), resize))

    x = np.array(x)
    x = x.reshape(len(x), -1)
    x = x / 255

    model = load_model('finalproject.hdf5')
    y_pred = np.argmax(model.predict(x), axis=1)
    return y_pred, model
```

4. Result

Painting Analysis





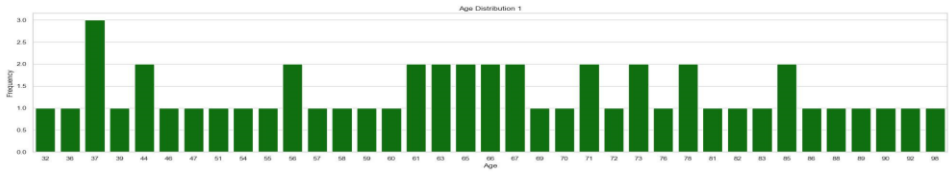
Painting Distribution:	
count	50.000000
mean	168.920000
std	157.451105
min	24.000000
25%	81.000000
50%	123.000000
75%	191.750000
max	877.000000

Plot 2.1-A: Painting Distribution Plot **Plot 2.1-B: Painting Distribution ViolinPlot** **Table2. 1: Descriptive Statistics of paintings**

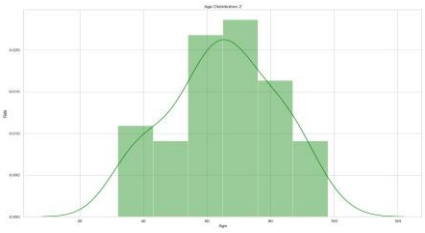
From the above two plots of painting distribution and descriptive statistics, it is clear that artists created 168 paintings averagely in their lifetime. The bigger number of paintings an artist created is 877 and the least is 24.

Sorting by the number of paintings in descending order, we pick up the top 10 artists who created the most paintings for the machine learning process. The top 1 artist with the most paintings is the most famous and influential figures in Western art, Dutch post-impressionist painter Vincent van Gogh. His paintings like landscapes, still life's, portraits and self-portraits, and are characterized by bold colors and dramatic, impulsive and expressive brushwork that contributed to the foundations of modern art.

Age Analysis



Plot: Age Frequency Plot

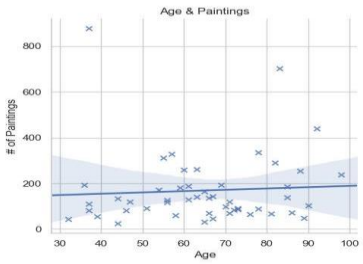


Age Distribution:	
mean	64.78000
std	16.74087
min	32.00000
25%	55.25000
50%	65.50000
75%	77.50000
max	98.00000

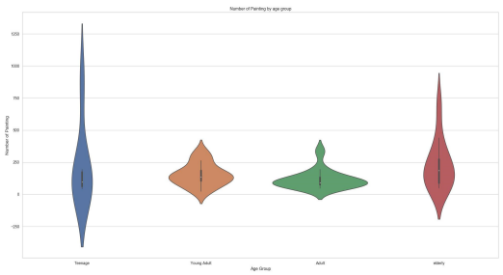
Plot: Age Distribution Plot **Table: Descriptive Statistics of age**

The distribution of artist age is nearly normal distribution given the above plots. The oldest artist was 98-year-old and one artist passed away at the young age of 32. We could create a new variable

age_group based on the descriptive statistics table. However, we will define the age breakpoint based on social conventions.

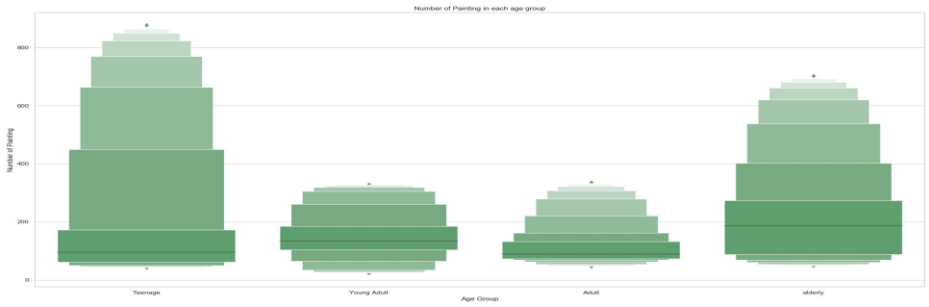


Plot: Age VS. Paintings



Plot: Paintings Distribution by Age Group

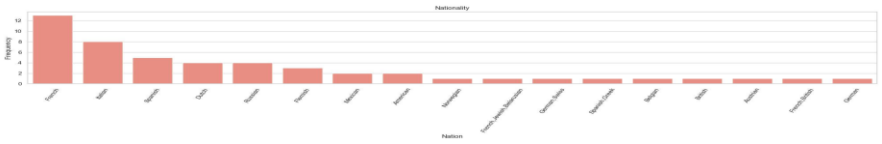
Plot 2.2C shows a simple linear regression of painting and age. The longer lifetime was, the more painting an artist could create, which is very straightforward. There are several outstanding outliers like Dutch artist Vincent van Gogh who created 877 paintings in a lifetime of 34 years, and French artist Edgar Degas creating 702 paintings in his 83 years lifetime. We will keep all for models.



Plot: Paintings Frequency by Age Group

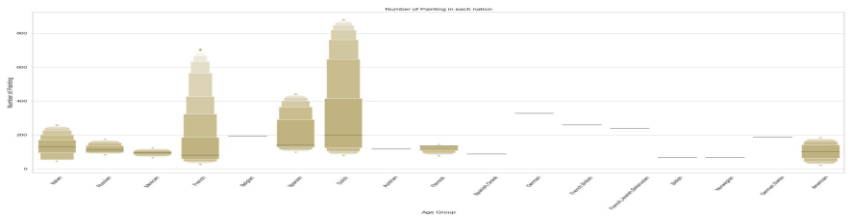
Plot 2.2 C and D are violinplot and boxenplot of painting distribution in each age group respectively. Both plots indicate that the range of painting numbers cross widely in senior and young adults' groups, which is definitely the effect made by the two famous artists Vincent van Gogh and Edgar Degas we mention before.

Nation Analysis



Plot: Nation Frequency

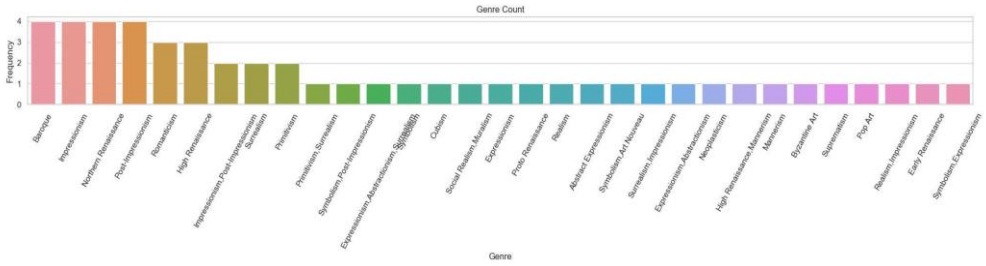
The top 3 countries with the most famous artists are French, Italian and Spanish, in such order. All are in Europe. It is easy to understand given the history of Renaissance art and painting, produced during the 14th, 15th, and 16th centuries in Europe under the combined influences of an increased awareness of nature, a revival of classical learning, and a more individualistic view of man.



Plot 2.3-B: Paintings by Nation

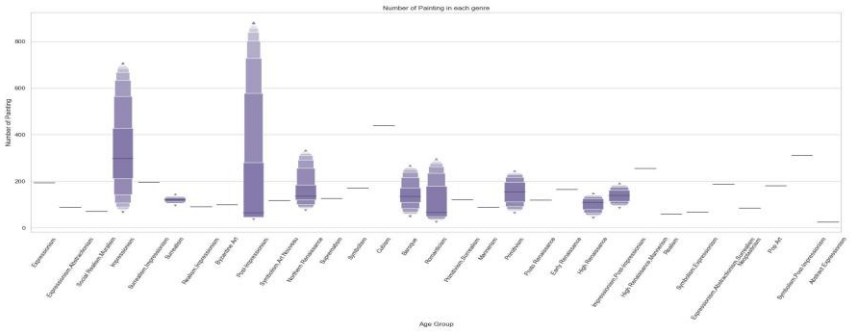
The boxenplot proves what we find out from the frequency table. French, Italian and Spanish have more artists than other counties. The outlier of Vincent van Gogh and Edgar Degas stand out in this plot as well.

Genre Analysis



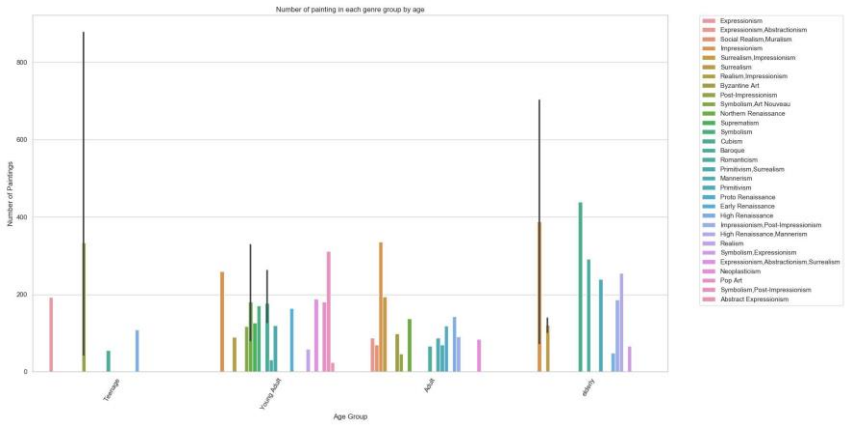
Plot: Genre Frequency

Here are the artists counted by genre. It is nice to see that we have so many different kinds of art genre count up to 24. There are 4 artists in Baroque, Impressionism, Northern Renaissance and Post-Impressionism each.



Plot: Paintings by Genre

Vincent van Gogh was a Dutch post-impressionist painter who is among the most famous and influential figures in the history of Western art and created 877 oil paintings, most of which date from the last two years of his life. So, genre of post-impressionist in this boxenplot is the tallest. The second tallest bar is the genre of Impression, which has something to do with Edgar Degas who we keep mentioning. Edgar Degas was a French impression artist famous for his pastel drawings and oil paintings of ballerinas.



Plot: Number of Painting by Genre in each Age Group

There are way more different kinds of genre among the young adult and adult artist groups, compared to teenage and elderly groups. The most popular genres of impression and Baroque could be found in all 3 age groups except teenagers.

5. Summary and Conclusion

The Keras MLP model I want to use is from Exam 1. Given the unbalanced dataset which contains paintings from 10 artists, the preprocessing should include data augment step to balance the dataset.

The model result was not good when first running before, so I did not keep a record of the bad result. Later I updated the PyCharm to version 3.4 and all sudden there are some problems related to VM instance setting and code could not find the directory while running. So, this model could not run successfully in the end and was not used in the final presentation. I will reset the instance in AWS instead of GCP and work on the model later.

6. Percentage of code

Percentage: 37%

7. References

https://en.wikipedia.org/wiki/Multilayer_perceptron

https://keras.io/applications/#resnet

https://keras.io/preprocessing/image/