Managing ML Data and Models for Painting Classification Final Report

Zihan Qin, Jiaqi Wu

University of Southern California

1. Background

With the development of the Internet, people now can easily appreciate artworks by browsing online artwork websites (e.g. WikiArt[1]). These artwork websites present paintings' introduction, making people know about these paintings' information (e.g. style, genre, artist). However, only if a person searches for a painting's name can he or she get its information. If people see pictures without any introduction and want to know about their information like their style or genre, they cannot find what they need by putting a photo of the painting into an artwork website's search engine. To solve this problem, we plan to develop a system that can predict a painting's artist, style, and genre when the user input a picture of a painting. To be more specific, our system aims to use several classification models trained on the existing artwork dataset and provide a user-friendly web page for users to browse the paintings in the existing artwork dataset and get information about a new painting. We hope our system can help people appreciate artworks in a more convenient way.

2. Description

Our system contains several important parts including data storage, database construction, feature extraction, model training and Web development. The dataset we use in this project is WikiArt dataset and is stored on Amazon S3. The features are extracted by Spark and stored in the database on Amazon RDS with data's other information. The SVM classification model used in this system is trained on the WikiArt dataset and stored in Amazon S3 as well. An user-friendly web page is realized based on Django and Bootstrap.

[1] Visual art encyclopedia. www.wikiart.org. (n.d.). Retrieved September 17, 2021, from https://www.wikiart.org/.

3. Architecture

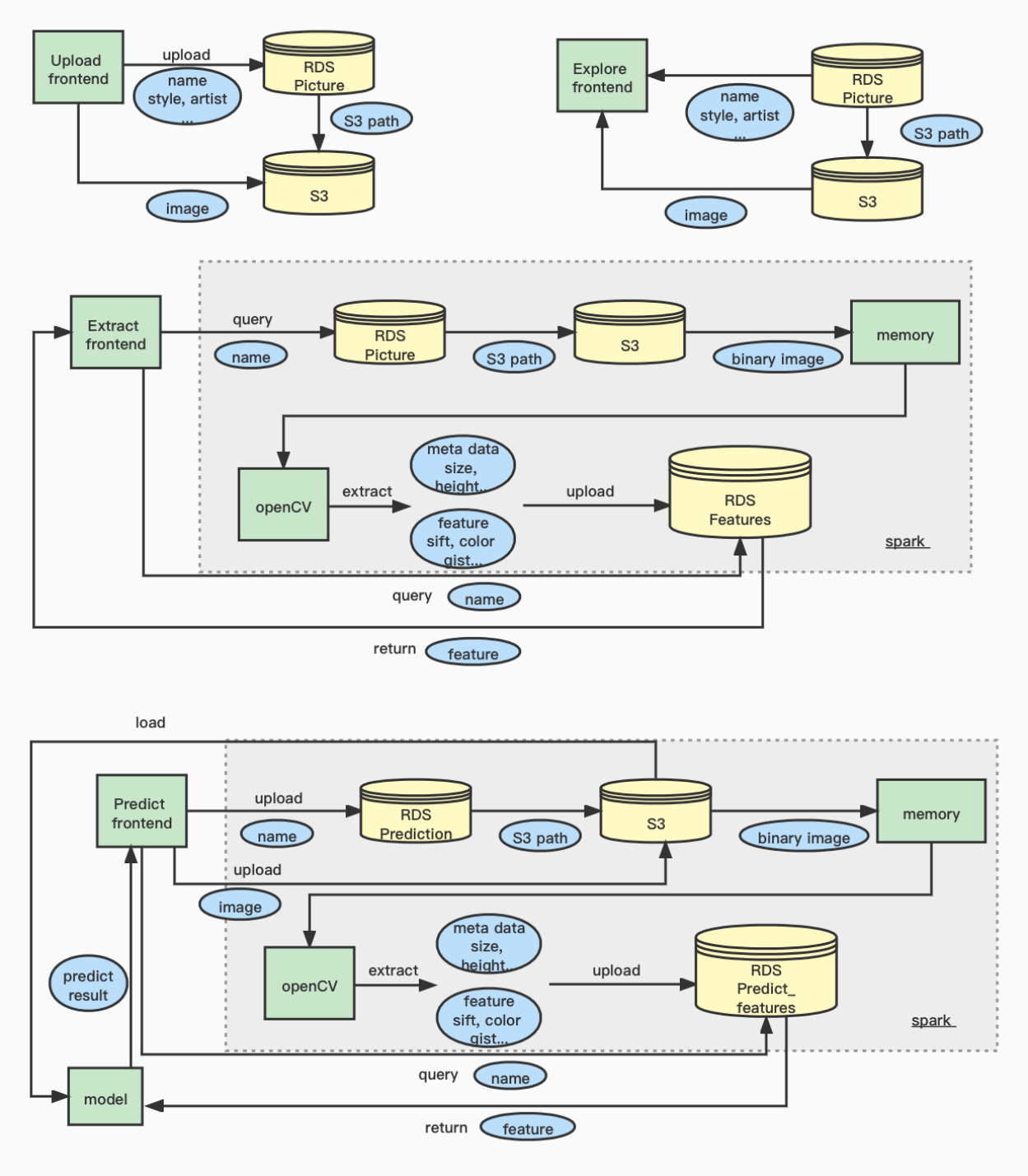

Figure 1. System architecture

4. Components

4.1 Cloud Storages

We use free cloud storages created by Amazon to support the data storage demand of

this project. There are two types of cloud storages in our application, which is AWS S3 and AWS RDS.

4.1.1 AWS S3

S3 is the abbreviation for Amazon Simple Storage Service, which can store and protect a large amount of unstructured data in various formats. S3 stores train pictures dataset, machine learning models, and the pictures uploaded by users. The total size of S3 is 9.3 GB currently.

The directory structures in S3 are organized as only one bucket with two directories, one for storing raw images and one for model checkpoints. The bucket of raw images contains training samples and is also used for storing samples uploaded by users. The images are separately saved in different directory of styles, which makes use easier to extract all images of certain style. All paintings are stored in .jpg formation.

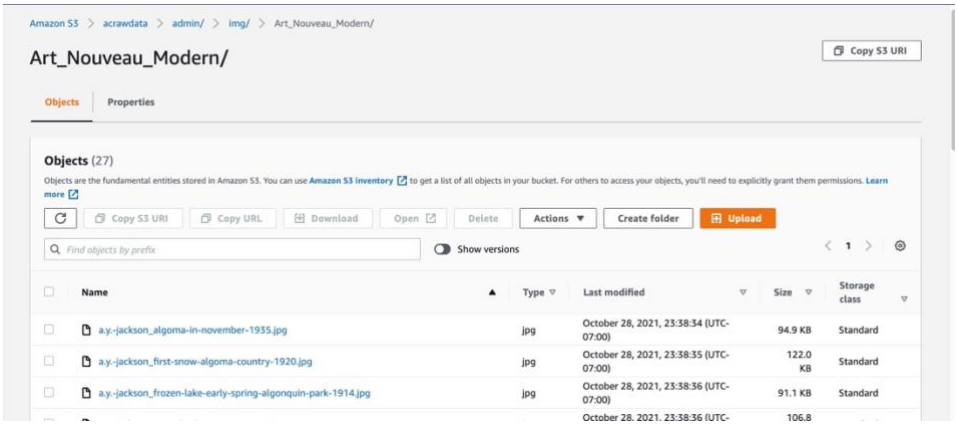

Figure 2. An illustration of S3 data. This figure shows raw images in the directory Art_Nouveau_Modern, a style of paintings. The formation of images is jpg. Types, last modified time, file size and storage class can also be checked in the portal of S3.

Files on S3 are not publicly accessible for safety reasons. The application uses unique access key id and secret access key to access the raw images on S3, which makes contributions to protect models and user images. With this key authorization, images in S3 can be obtained by http addresses. This http address is unique for each image, which can be considered as the signature of a painting. We use package boto3 in python to connect with AWS S3 and upload images from local machine. When uploading the images, boto3 will return the http address of it. The return addresses are stored in our AWS RDS database.

4.1.2 RDS

RDS is short for Amazon Relational Database Service, a web service that enables customers to efficiently set up and operate relational databases in the AWS cloud. RDS stores features, metadata, prediction results and other utilization information of web service.

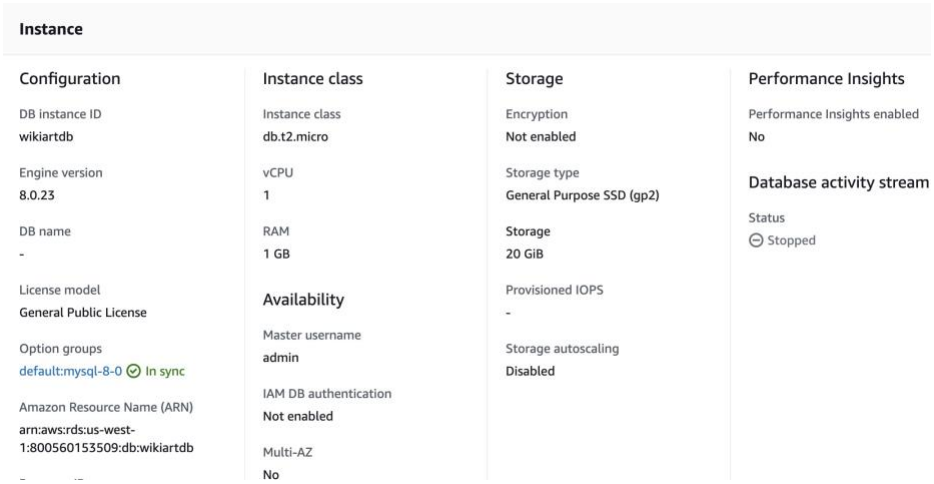

Figure 3. Configuration of our MySQL instance in AWS RDS.

AWS RDS database in our project is a MySQL database supported by AWS RDS. We use this database to store metadata of raw images, features and other user information. The two main tables in this MySQL dataset are table *Picture* and table *Features*. The detailed structure of these two tables will be showed in the 'Web Development' section.

Our MySQL database on AWS RDS is publicly accessible. However, the username and password are still needed for access and modify the data. We use MySQL connectors in Python and Java to create interfaces for creating and modifying tables, inserting, and reading the data with the same query in MySQL.

4.2 Feature Extractor

4.2.1 Tools

We use Spark to build the feature extraction pipelines and employ OpenCV and other third-party tools to extract features from raw images. Spark provides tools for connecting Amazon cloud storages and parallel solutions of image processing, which significantly

improve the efficiency of our pipeline.

The script of Spark includes following steps:

Connecting to MySQL in AWS RDS.

We connect to MySQL with the java database connector (jdbc) in Spark and download the photos_figure tables as Spark DataFrame.

2. Extracting http addresses and changing it to S3 addresses

The S3 http paths of train images are collected and filtered. Then, they will be changed into S3A format for spark to access the images on S3.

3. Accessing AWS S3 and downloading raw images

We access AWS S3 in Spark with hadoop-aws tools and download the target images in OpenCV image formation.

4. Generating features

More details about algorithms are described in Section 4.2.2

5. Writing features into MySQL in AWS RDS

Finally, we write features into a Spark Dataframe and dump it to MySQL table photos_features by jdbc.

4.2.2 Algorithms

We extract three types of features from the images, which are SIFT, color moments and GIST features. Besides, OpenCV is employed for extracting metadata of images, such as height, width, depth and mode.

SIFT feature

SIFT is short for the scale-invariant feature transform, which is one of the most popular techniques used in detecting, describing and matching images. A series of SIFT key points (we denote as SIFT features in our project) are extracted first. Then the similarity of images can be described as the Euclidean distance between their SIFT key points. We use SIFT algorithm in OpenCV to generate the SIFT key points of our images.

Color moments feature

Color moments are designed for describing the color distribution of an image. As central moments in possibility, color moments in different orders contain different information about the colors on images. Color moments can be used to identify similar images from the perspective of colors. We simply include the first to third order moments as our features, which is calculated with OpenCV.

GIST feature

GIST features are global image features in low dimension representation. It basically is constructed with convolution and whitening techniques. It summarized the gradient information for different parts of images. The cosine similarity between GIST features are proved to be a strong factor of image similarity. We use a third-party python package Pyleargist to extract GIST features from images.

4.3 Model

For prediction, we use SVM machine learning model and setting parameters by Grid Search.

4.3.1 SVM

SVM is short for support-vector machines, which maps training examples to points in high-dimensional feature spaces by Radial Basis Functions (RBF) and find the maximum margins among categories. For prediction, the new examples are mapped into the same space and allocated to classes based on the trained margin.

Two hyperparameters mainly affect the model performance. Gamma parameter defines the influence of a single training example. The lower value gamma is, the further influence single examples have. C parameter represents the balance between margin maximization and correct classification of training examples. The low-value C weight large margin more than training accuracy. We search the appropriate parameters for models of different tasks by Grid Search:

Prediction task	highest accuracy of 5-fold	accuracy on test set
Style	0.540	0.58
Genre	0.503	0.55
Artist	0.373	0.42

Table 1. Performances of prediction models

Due to the time limit of this project, the prediction accuracy has not reached the upper limit. There are several ways for further improvement:

(1) Model: Other machine learning or deep learning models can be tested on these tasks.

(2) Dataset: The training dataset is unbalanced among classes. Although we have tried some basic methods to remedy class imbalance, we think that some better balancing treatments may improve the prediction results more.

5. Web Development

5.1 Database

The system is connected directly to Amazon RDS MySql by Django's Settings. Seven models are created and stored in Amazon RDS as tables for realizing all the functions.

(1) Genre

Column	Type	Description
id	bigint	An unique representation of one genre (primary key)
name	Varchar(100)	The genre's name, e.g. portrait, landscape…

Table 2. Detailed structure of table Genre

This table is created by the labels of training data.

(2) Style

Column	Type	Description
id	bigint	An unique representation of one style (primary key)
name	Varchar(100)	The style's name, e.g. Baroque, Impressionism…

Table 3. Detailed structure of table Style

This table is created by the labels of training data.

(3) Artist

Column	Type	Description
id	bigint	An unique representation of one artist (primary key)
name	Varchar(100)	The Artist's name

Table 4. Detailed structure of table Artist

This table is created by the labels of training data.

(4) Picture

Column	Type	Description
name	Varchar(512)	An unique name of the artwork (primary key)
image	Varchar(100)	It is an imagefield defined in django.models and records the image's path in S3 (e.g. /**/**.jpg); the corresponding image file is stored in Amazon S3
artist_id	bigint	The foreign key referring to Artist table
genre_id	bigint	The foreign key referring to Genre table

style_id	bigint	The foreign key referring to Style table
url_path	Varchar(512)	The url path of the corresponding image file

Table 5. Detailed structure of table Picture

This table is created by the training data

(5) Features

Column	Type	Description
id	bigint	An unique representation of this fetuare
height	bigint	Height of the image (metadata)
width	bigint	Width of the image (metadata)
nChannels	bigint	The image's number of channels (metadata)
mode	bigint	The image's mode (metadata)
sift	longtext	SIFT feature of the image (feature)
color_moments	longtext	Color moments of the image (feature)
color_gist	longtext	Color GIST of the image (feature)
image_id	Varchar(512)	The foreign key referring to Picture table

Table 6. Detailed structure of table Features

This table is created by the features of training data.

(6) Predict_option

Column	Type	Description
id	bigint	An unique representation of one option (primary key)
name	Varchar(100)	The predict option's name, e.g. style – means predict style

Table 7. Detailed structure of table Predict_option

Due to our objective, three predict options are stored in this table.

(7) Prediction

Column	Type	Description
id	Varchar(512)	An unique identification of a prediction record
image	Varchar(100)	The image for prediction
predict_result	longtext	classification result of this image
predict_option_id	bigint	The foreign key referring to Predict_option table, determining the predict task
url_path	Varchar(512)	The url path of the corresponding image file

Table 8. Detailed structure of table Prediction

Prediction records are generated and stored in this table.

(8) Predict_features

Column	Type	Description
id	bigint	An unique representation of this fetuare
height	bigint	Height of the image (metadata)
width	bigint	Width of the image (metadata)
nChannels	bigint	The image's number of channels (metadata)
mode	bigint	The image's mode (metadata)
sift	longtext	SIFT feature of the image (feature)
color_moments	longtext	Color moments of the image (feature)

color_gist	longtext	Color GIST of the image (feature)
image_id	Varchar(512)	The foreign key referring to Picture table

Table 9. Detailed structure of table Predict_features

This table has the same structure as the features table above, but it is used for storing features of images for prediction.

5.2 System Function Realization

This system's functions include exploring artworks, uploading a new artwork to the database, extracting metadata and features, predicting an artwork's style, genre or artist. How each function is realized will be described in detail below.

(1) Explore Artworks

At home page, artworks are displayed with their information. To realize this function, our group used boto3 package to connect to Amazon S3. As all the information needed for displaying is stored in the table *Picture*, the system can get all the objects in the table *Picture* from Amazon RDS and load images from S3 by the field *image* in *Picture* which records the file path on S3.

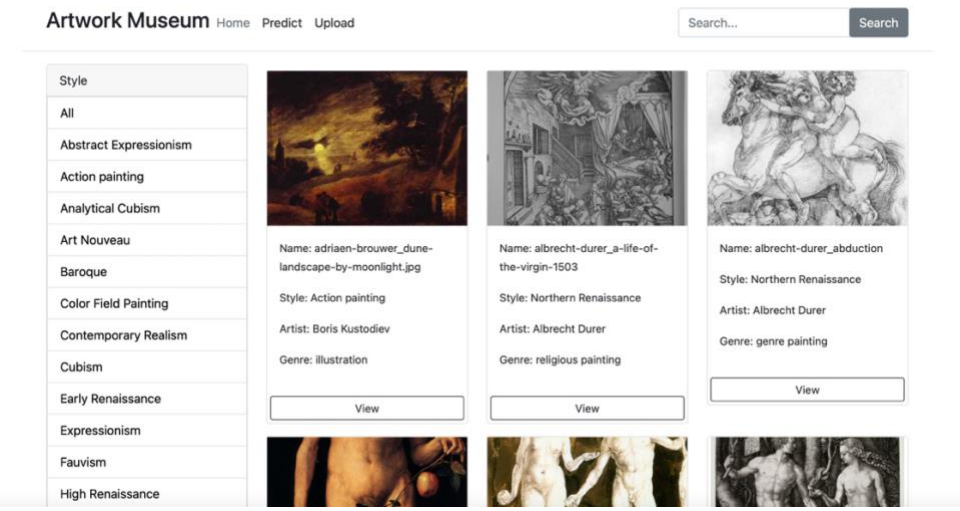

Figure 4. Homepage

To make exploring data more convenient, users can select specific style or search by a keyword. This function is realized by getting a style or a keyword from the interface and return it to the back-end. Then the back-end is able to filter the objects by the given information.

By clicking on button 'view', an user can view this artwork's page which includes its

complete picture, its introduction and its metadata and features. Its metadata and features are get from table *Features*.

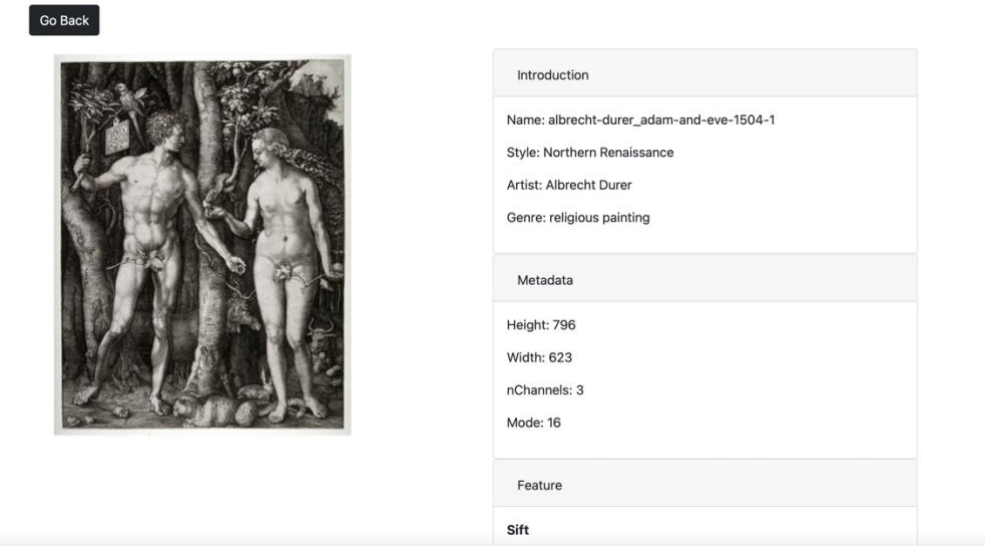

Figure 5. A sample artwork's page

(2) Upload Artworks

At upload page, the user can choose to upload an image of the artwork, input its name, style, artist and genre. After clicking on the submit button, this upload form would be sent to the back-end and used to generate a new data row in *Picture* table. The corresponding image file would be stored in Amazon S3.

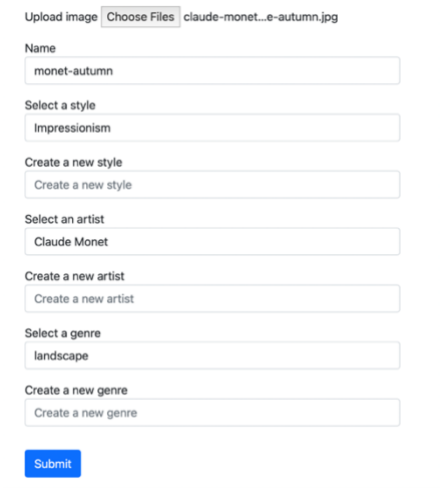

Figure 6. Upload page

(3) Extract Metedata and Features

Some artworks, like a newly uploaded artwork, don't have metadata and features. To extract an artwork's metadata and feature, an user can use the 'Extract Metadata and Features' button on an artwork's page. After clicking on this button, the back-end will get this artwork's primary key in table *Picture*, its name, and pass it as a parameter to the script for extracting metadata and features. By running the extracting script with Spark, all the metadata and features would be extracted and stored to the table *Features*. After that, the back-end can get this artwork's features from table *Features* and show them on this page.

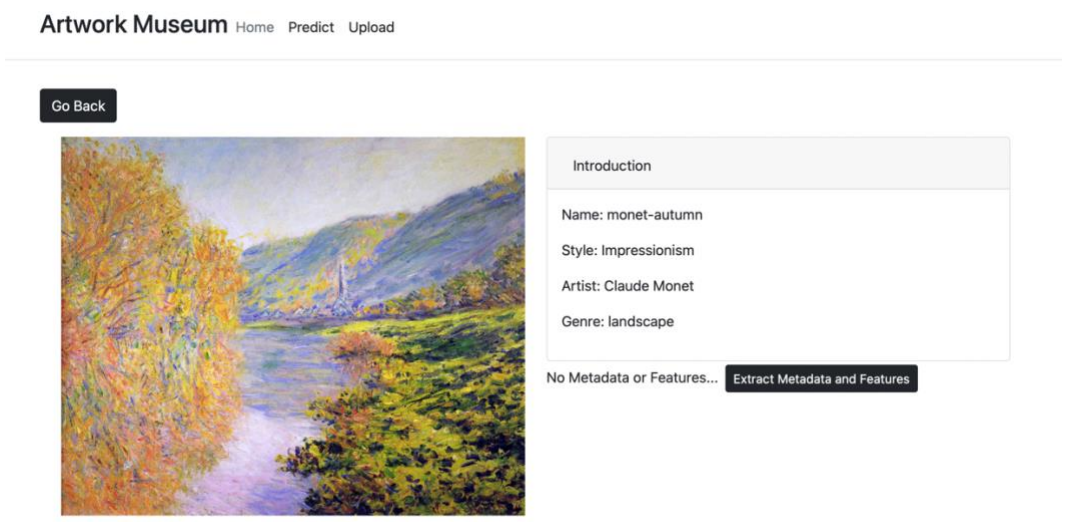

Figure 7. A sample of newly-uploaded artwork's page whose metadata and features hasn't been extracted.

(4) Predict

An user can use the predict page to predict the style, genre or artist of an artwork. This process is completed by firstly uploading the image for prediction and predict option to table *Prediction;* secondly, run the script for extracting metadata and features and add them to table *Predict_features*; then, load the prediction model from Amazon S3 according to the predict option and make a prediction based on this artwork's features which are get from table *Predict_features*; finally, add the predict result in table *Prediction* and display the most recent 10 records.

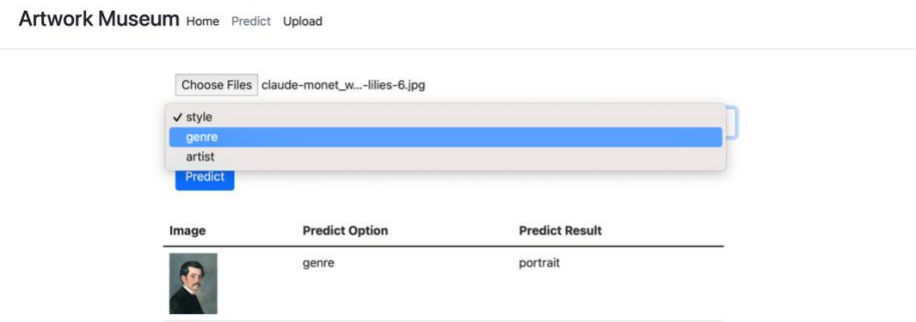

Figure 8. Predict page

6. Learning Experiences

Jiaqi Wu

By participating in this project, I have learned how to use Amazon S3 as storage in Django and how to manage image files for web development. Moreover, I have get more familiar with front-end and back-end development and how to design appropriate data structure in database for realizing specific functions. The biggest challenge for me during the whole process may be that I once failed to connect Amazon S3 with Django and met a lot of difficulties when trying to install all the required software for feature extraction, but finally, with my partner's help, I overcame all the obstacles and finished the project.

Zihan Qin

I have learned about implementing data analytics with cloud storage like Amazon S3 and RDS, which is a good way to store large amounts of data in a group project. I also learned the skill of Spark, speeding up the process of feature extracting on large amounts of data. Moreover, I have gained more experience in database structure design in a practice project. The biggest challenge for me is extracting pictures' features by spark. I have learned a lot from the internet and finally succeeded.

7. Project Members

Jiaqi Wu

Undergraduate Major: Data Science and Big Data Technology

Background Knowledge: Computer Vision; have experience in front-end and back-end

development

Current Program: Applied Data Science

Responsibility: Database Design; Model Training; Web Development

Zihan Qin

Undergraduate Major: Applied Statistics

Background Knowledge: Database, Statistics model

Current Program: Applied Data Science

Responsibility: Database Design; Data Preprocessing; Data ETL