The Red, The White And The Rosé

Final Project

Applied Machine Learning 2022

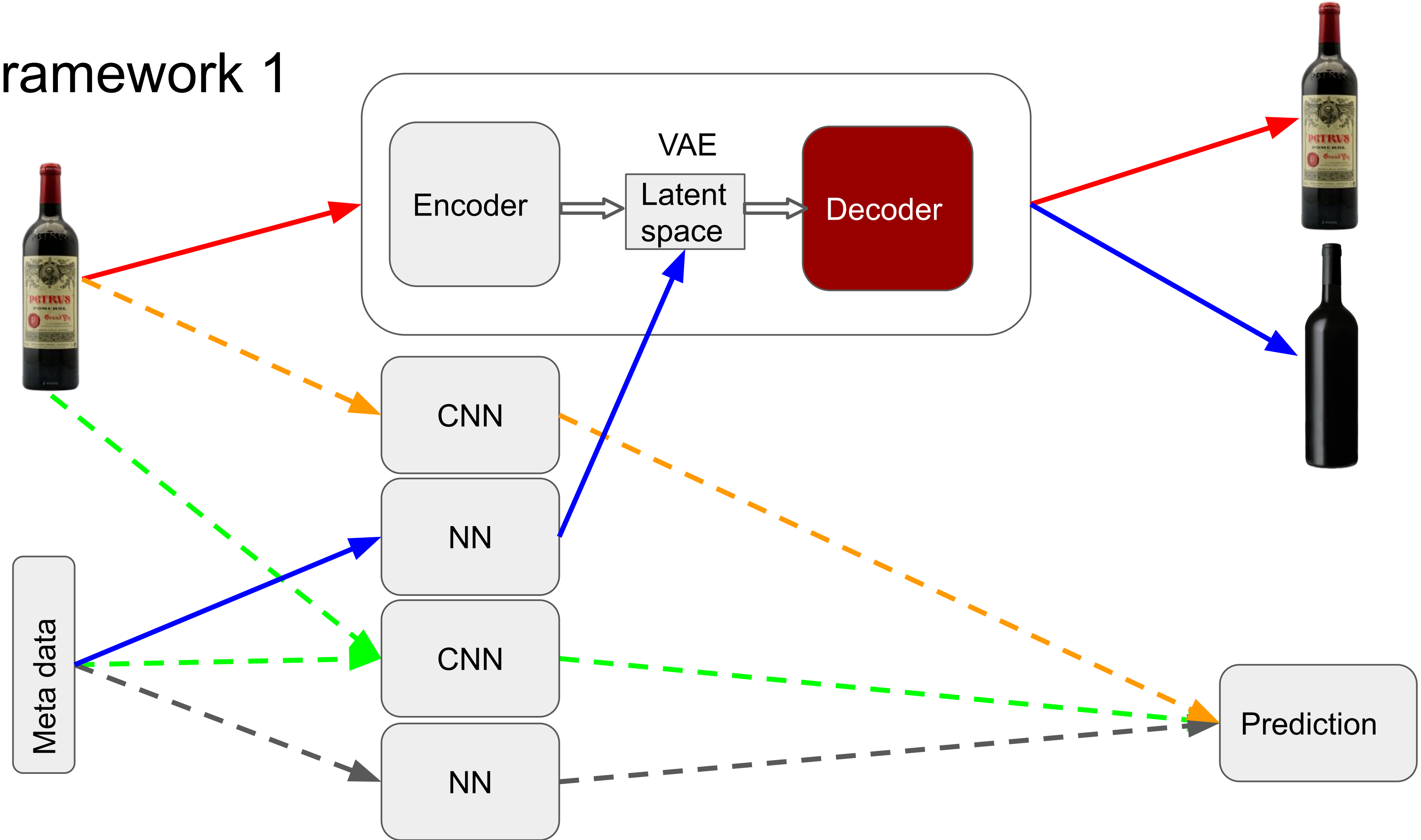
Norman Pedersen, Marcus Dreisler, and Steen Bender

Introduction

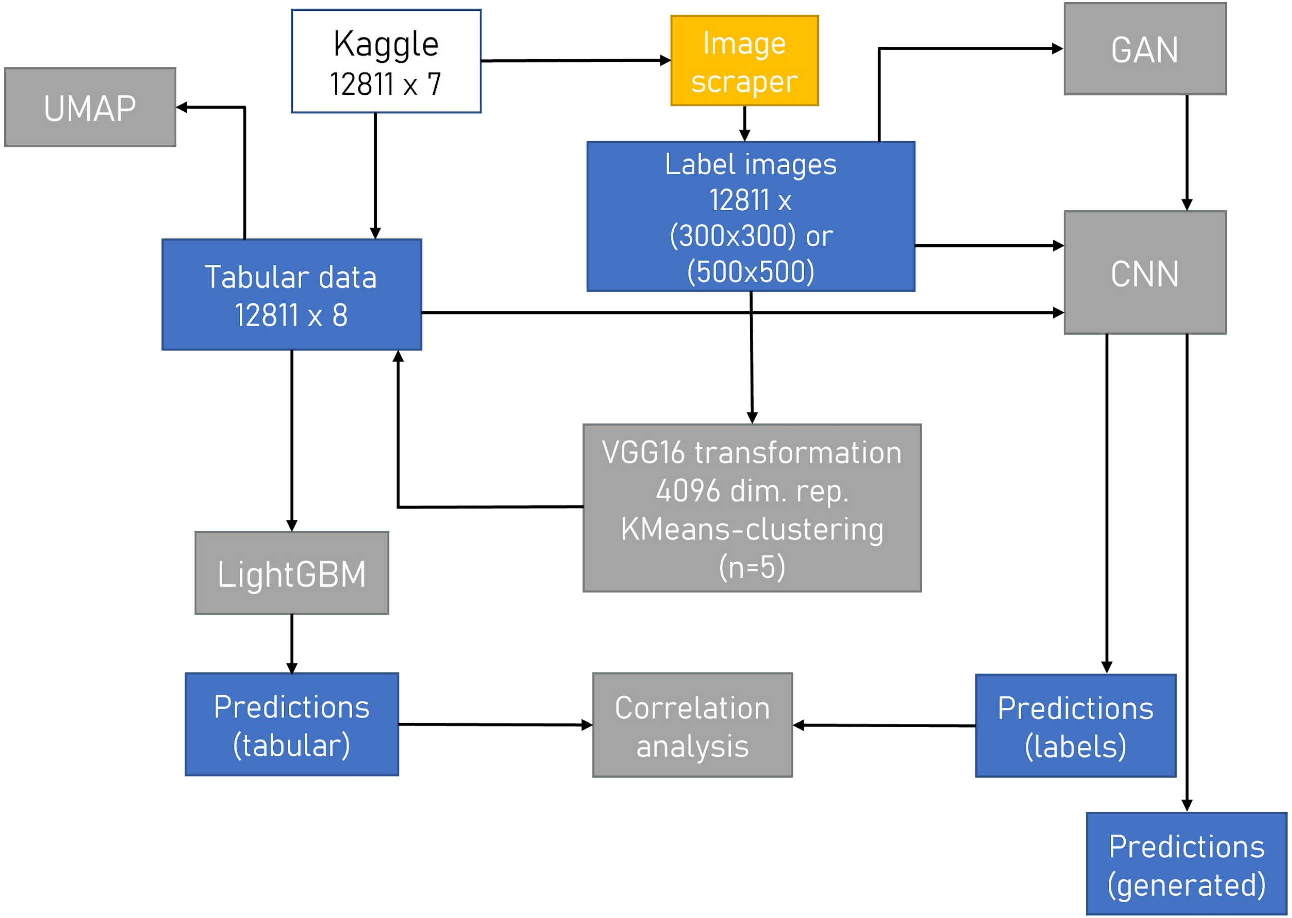
- Wine has been the subject in various studies - both in ML and other fields
    - Chemical analysis → assessing Quality & Rating
    - Written reviews → predicting Ratings
    - Wine metadata → predicting Ratings & Price

- We develop a ML framework to investigate the information in the bottle design and label (and hence visual consumer bias).

Framework 1


VAE
Encoder
Latent space
Decoder
CNN
NN
CNN
NN
Meta data
Prediction

Framework 2


Kaggle
12811 x 7
Image scraper
GAN
UMAP
Label images
12811 x
(300x300) or
(500x500)
Tabular data
12811 x 8
CNN
VGG16 transformation
4096 dim. rep.
KMeans-clustering
(n=5)
LightGBM
Predictions
(tabular)
Correlation
analysis
Predictions
(labels)
Predictions
(generated)

The dataset

Scraped from Vivino in 2020

13k unique wines with 8 variables

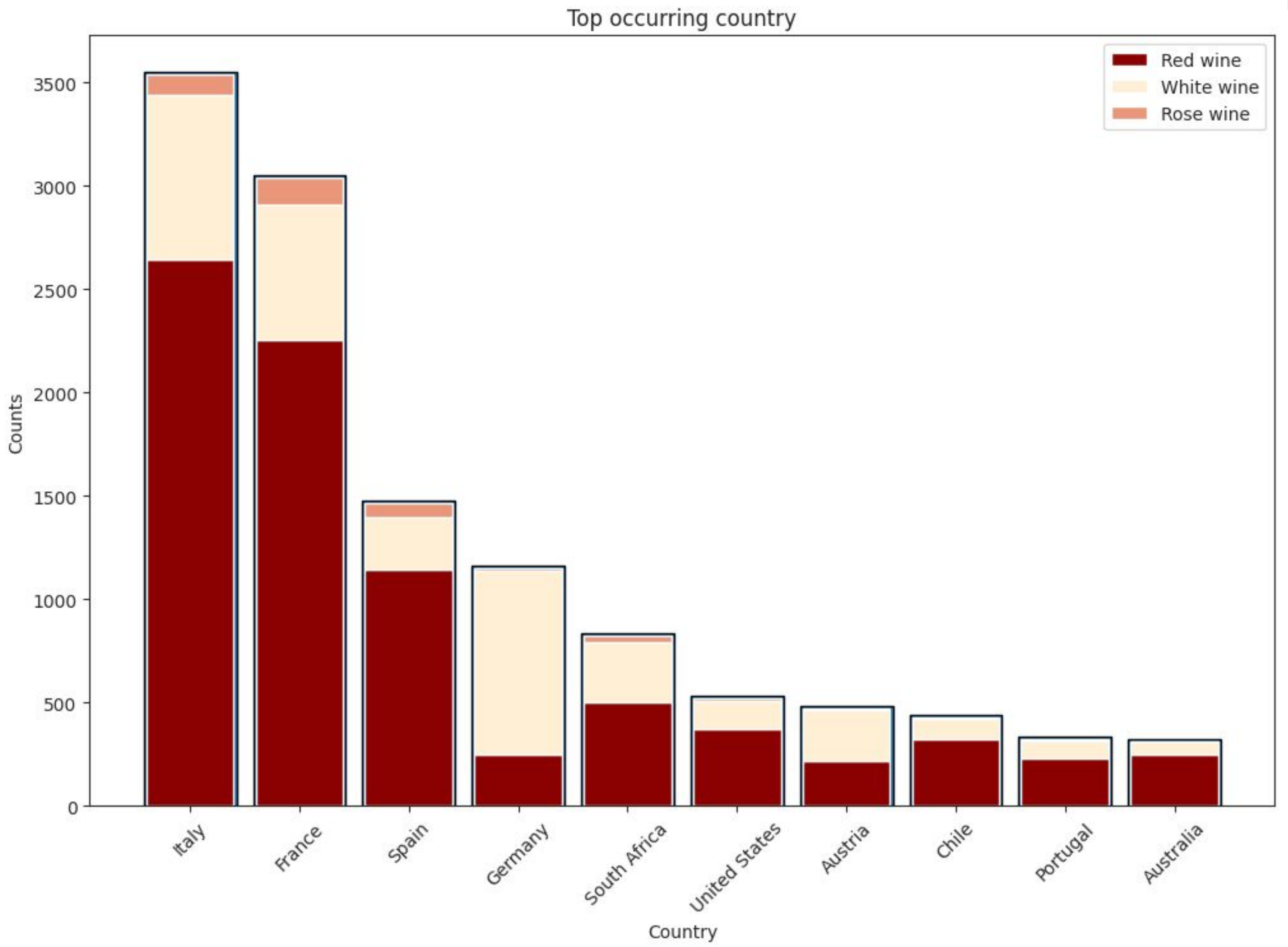
Name	Country	Region	Winery	Rating	No. ratings	Price	Year
Ribera del Duero	Spain	Ribera del Duero	Garmón	4.2	1017	38.80 EUR	2015

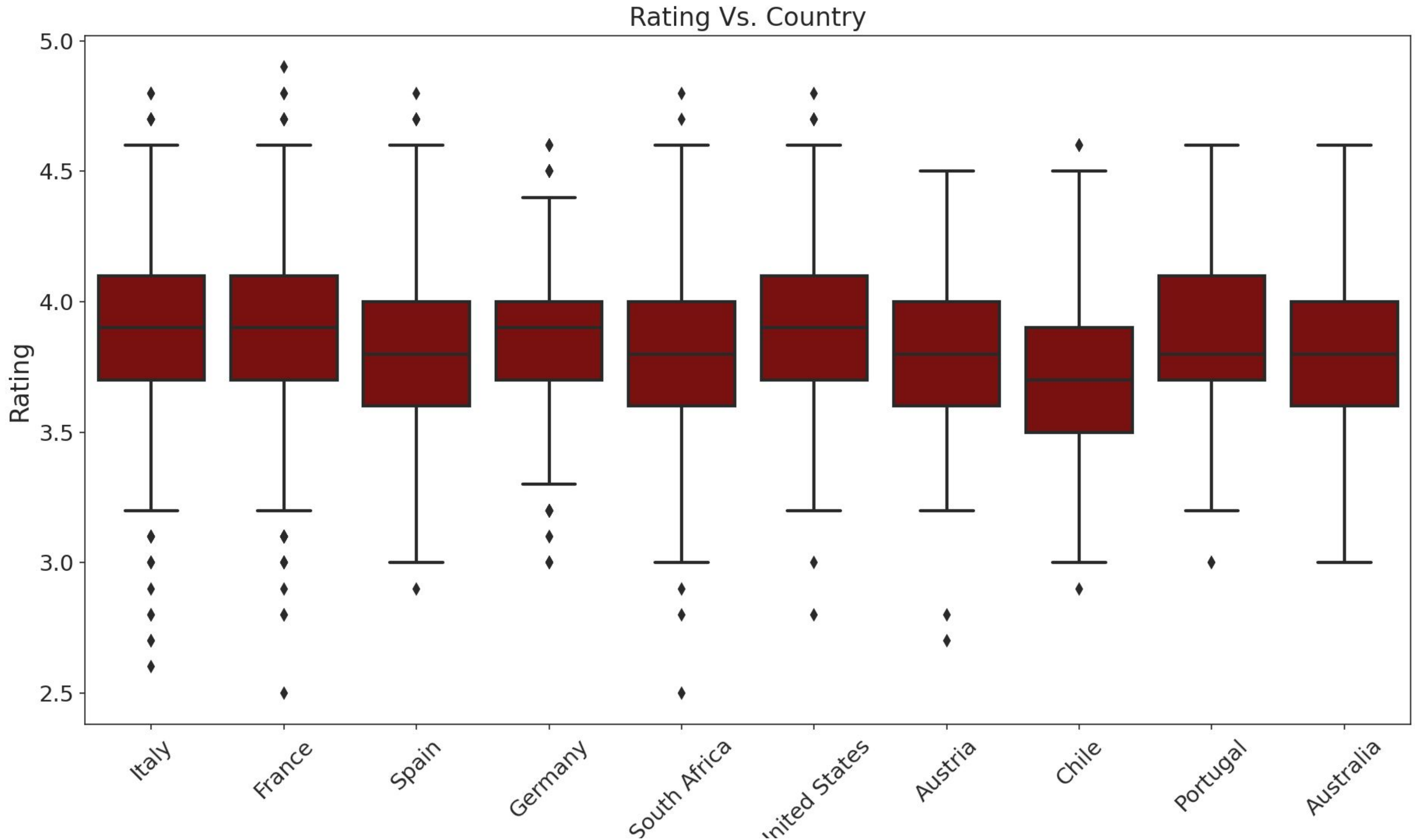
The dataset - an extended version

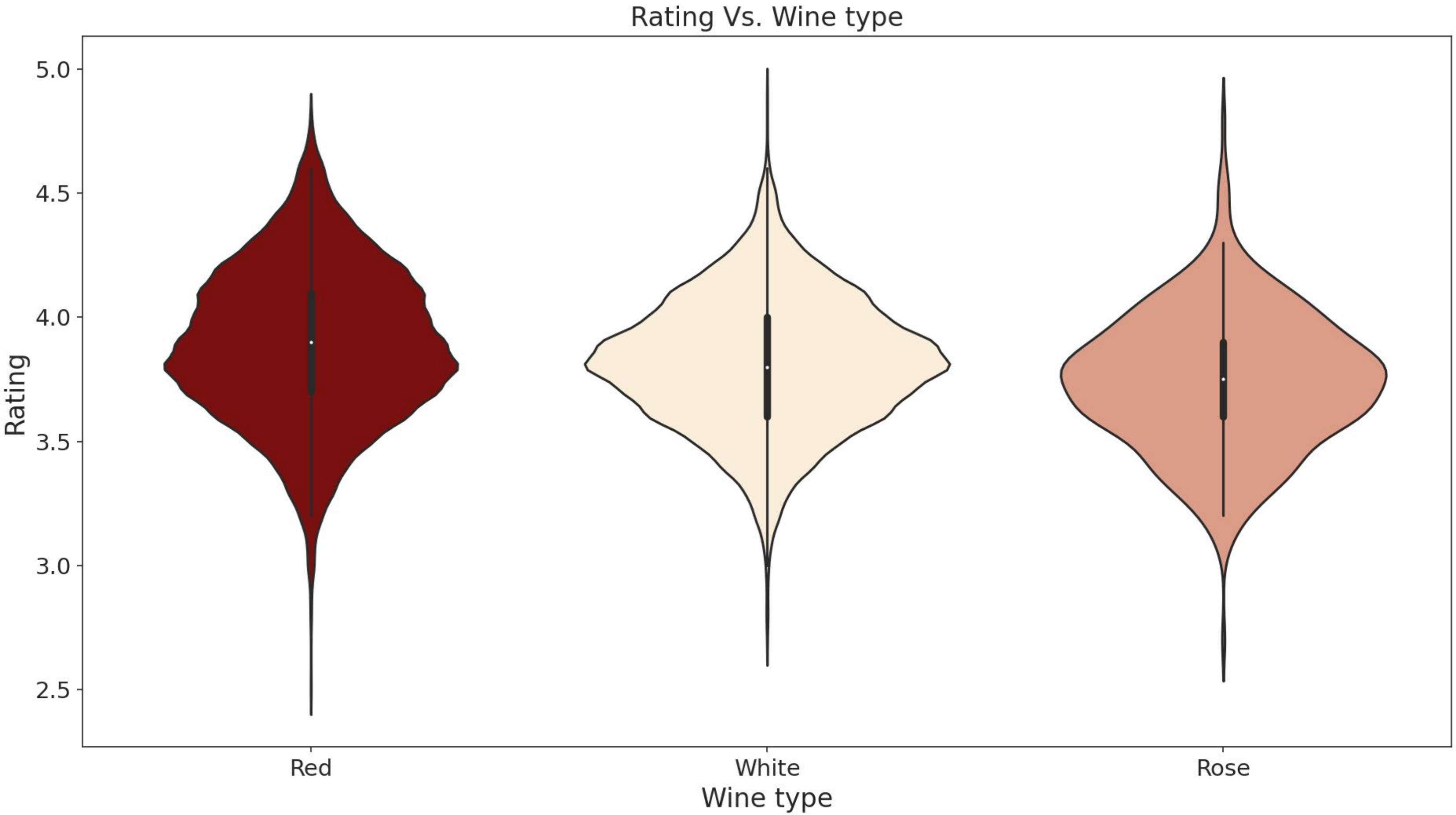
We made an additional scraper to obtain wine label images

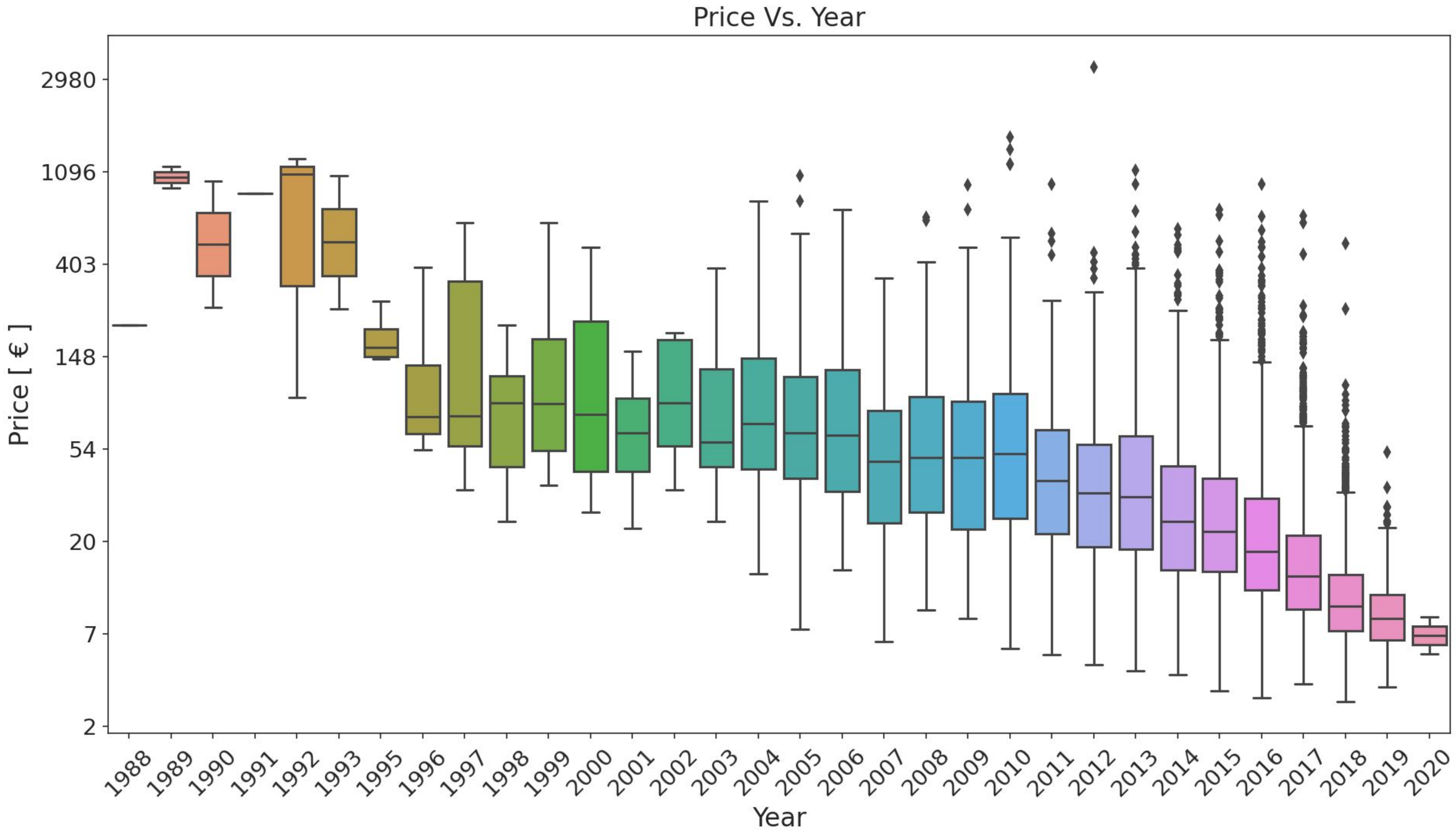
Google Maps API were also implemented to obtain *Latitude* and *Longitude* from *Winery, Region, and Country*

Name	Country	Region	Winery	Rating	No. ratings	Price	Year	Lat	Lng	Image
Ribera del Duero	Spain	Ribera del Duero	Garmón	4.2	1017	38.80 EUR	2015	42.45	-5.12	“Labels/Red_Wine/index.png”

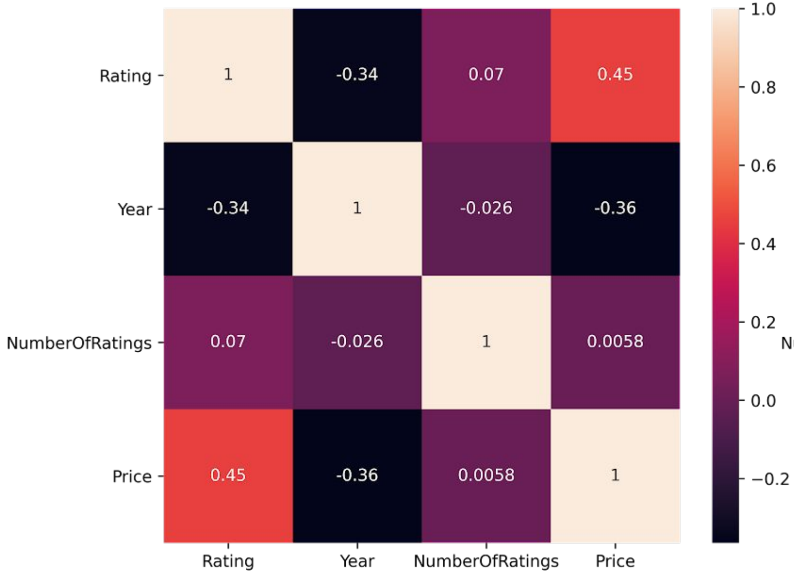

Top occurring country
Counts
Country
Red wine
White wine
Rose wine
0
500
1000
1500
2000
2500
3000
3500
Italy
France
Spain
Germany
South Africa
United States
Austria
Chile
Portugal
Australia

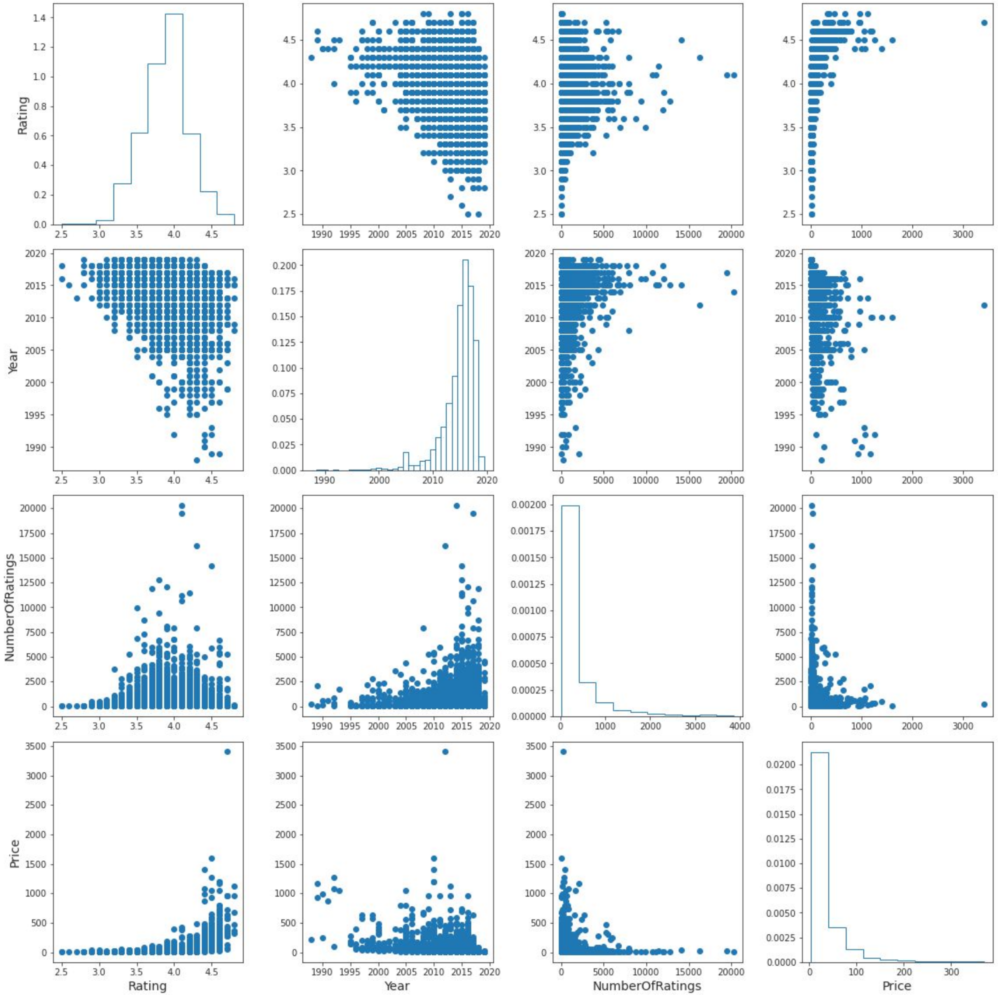

Rating Vs. Country
Rating
5.0
4.5
4.0
3.5
3.0
2.5
Italy
France
Spain
Germany
South Africa
nited States
Austria
Chile
Portugal
Australia


Rating Vs. Wine type
Rating
5.0
4.5
4.0
3.5
3.0
2.5
Red
White
Rose
Wine type

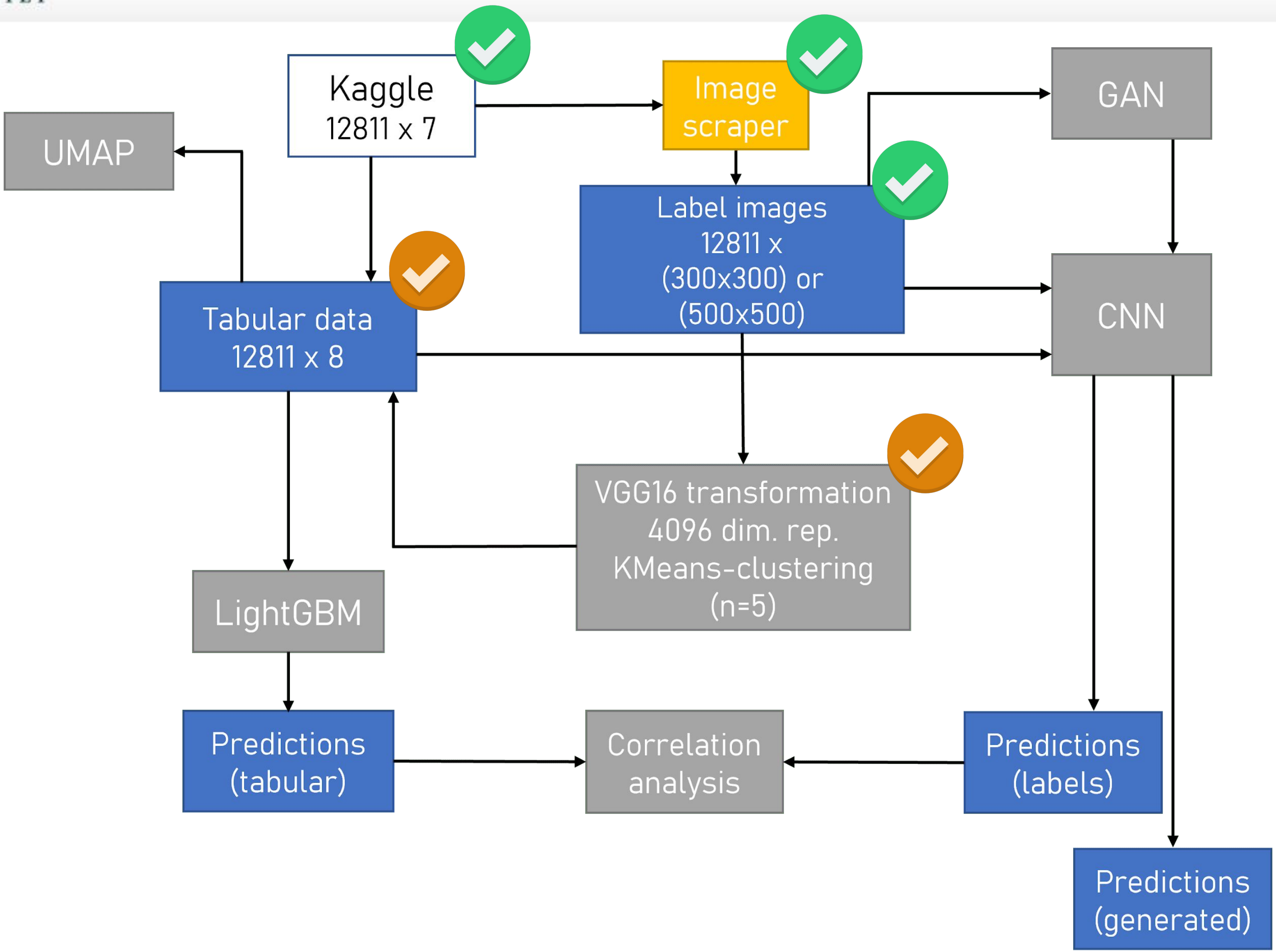

Price Vs. Year
2980
1096
403
148
54
20
7
2
Price [€]
1988
1989
1990
1991
1992
1993
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
Year

The raw numericals





(Red wine)

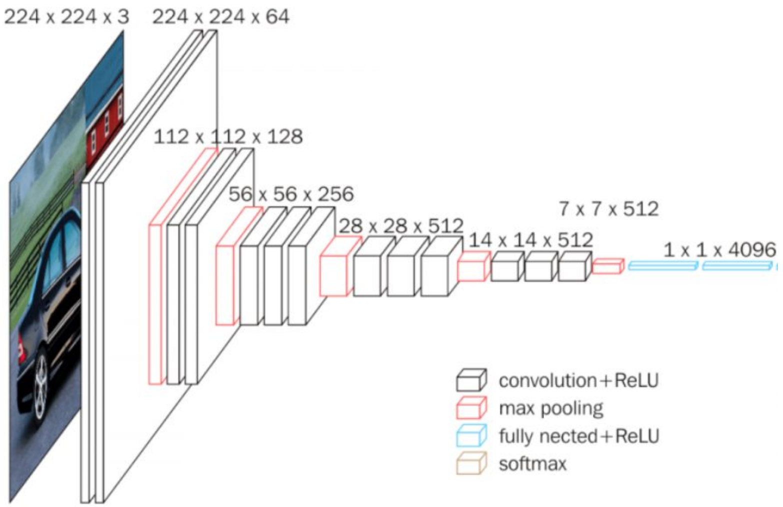

Kaggle
12811 x 7
Image
scraper
GAN
UMAP
Label images
12811 x
(300x300) or
(500x500)
Tabular data
12811 x 8
CNN
VGG16 transformation
4096 dim. rep.
KMeans-clustering
(n=5)
LightGBM
Predictions
(tabular)
Correlation
analysis
Predictions
(labels)
Predictions
(generated)

Separating hand-held and professional images

Transforming images using a pre-trained CNN, VGG16, to a 4096 dimensional representation

100 component PCA

K-Means clustering (n=5) of the 100 PCs



VGG16 by K. Simonyan and A. Zisserman (University of Oxford)

With this approach, we divide the wines into groups based on their label image.

The cluster integer further serves as a new variable, *img_cluster*, in the tabular data

Cluster int: 2

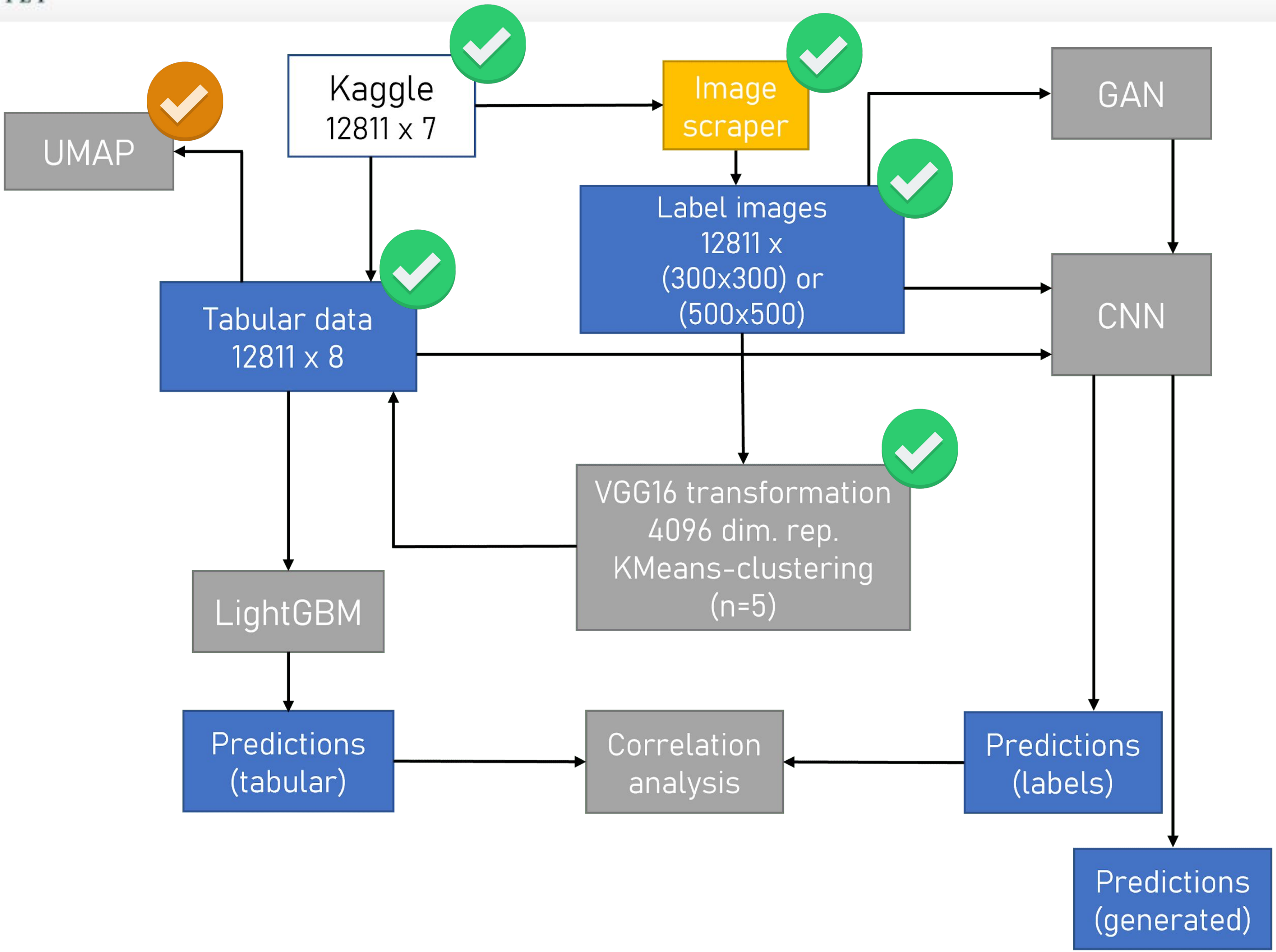
Cluster int: 1

Cluster int: 0

+ 2 more

Name	Country	Region	Winery	Rating	No. ratings	Price	Year	Lat	Lng	img_cluster
Ribera del Duero	Spain	Ribera del Duero	Garmón	4.2	1017	38.80 EUR	2015	42.45	-5.12	0

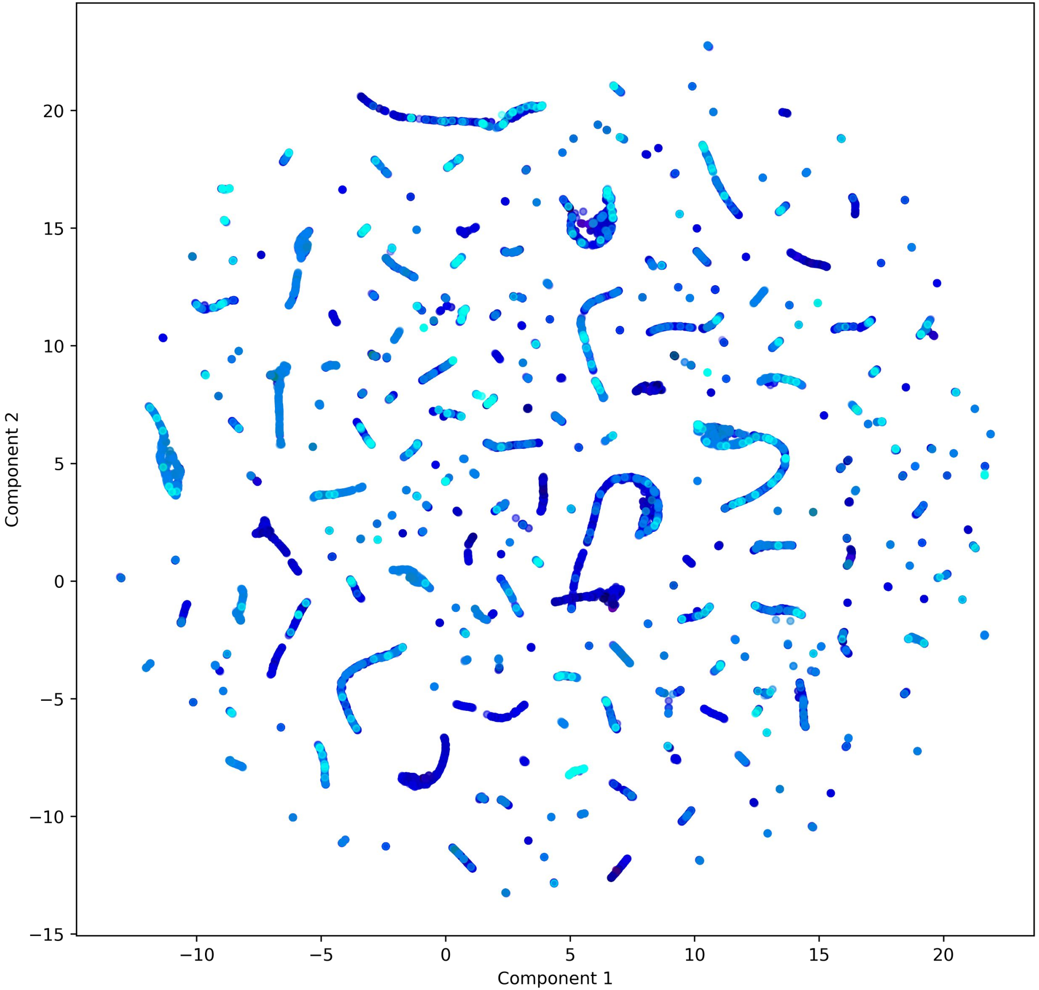
Strings


Kaggle
12811 x 7
Image scraper
GAN
UMAP
Label images
12811 x
(300x300) or
(500x500)
Tabular data
12811 x 8
CNN
VGG16 transformation
4096 dim. rep.
KMeans-clustering
(n=5)
LightGBM
Predictions
(tabular)
Correlation
analysis
Predictions
(labels)
Predictions
(generated)

“I always run a clustering algorithm at first…”

UMAP (2 components, 20 neighbours)

Colored by RGBA of price, type, year, and rating

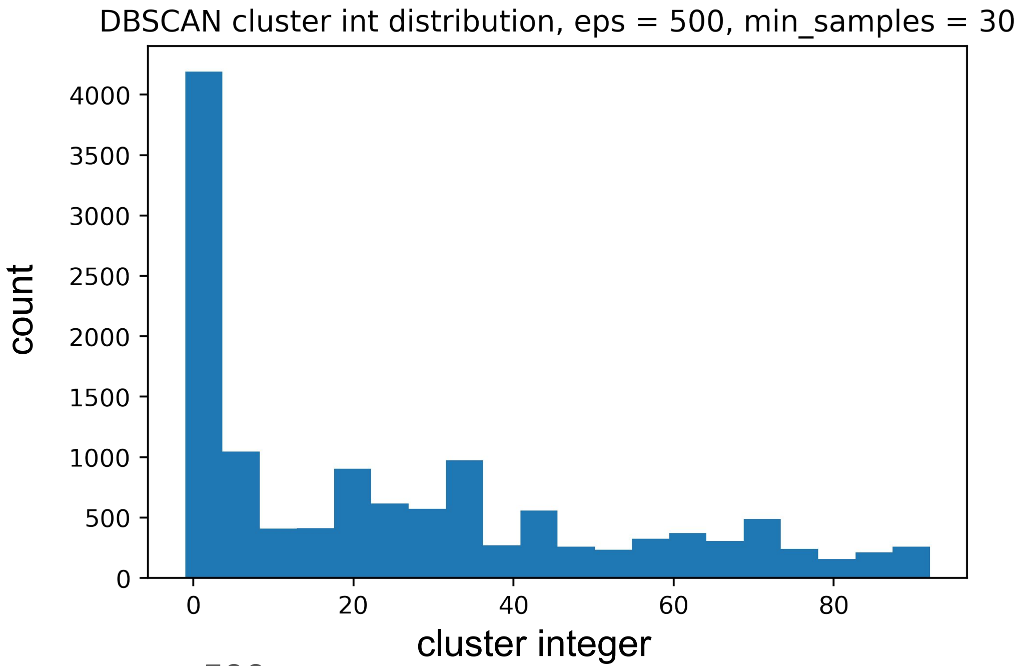


Vars: Price, Year, type_int, MercLat, MercLng, img_cluster, Rating, NumberOfRatings

DBSCAN

"Price",
"Year",
"type_int",
"MercLat",
"MercLng",
"img_cluster",
"Rating",
"NumberOfRatings"

94 clusters



eps = 500

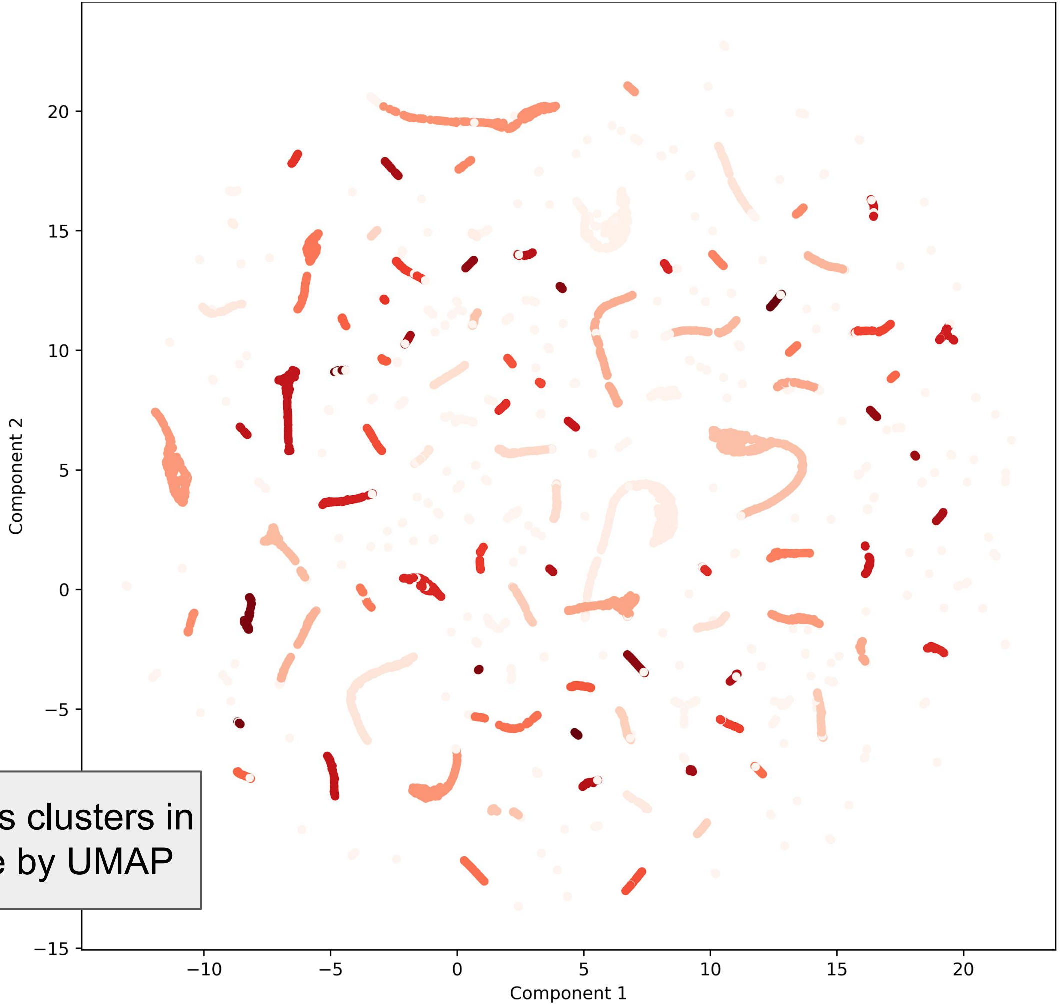
A high HP value (as compared to default), required to cluster the broadly distributed points in high dimensions

min_samples = 30

MinMaxScaler was not employed

UMAP (2 components, 20 neighbours)

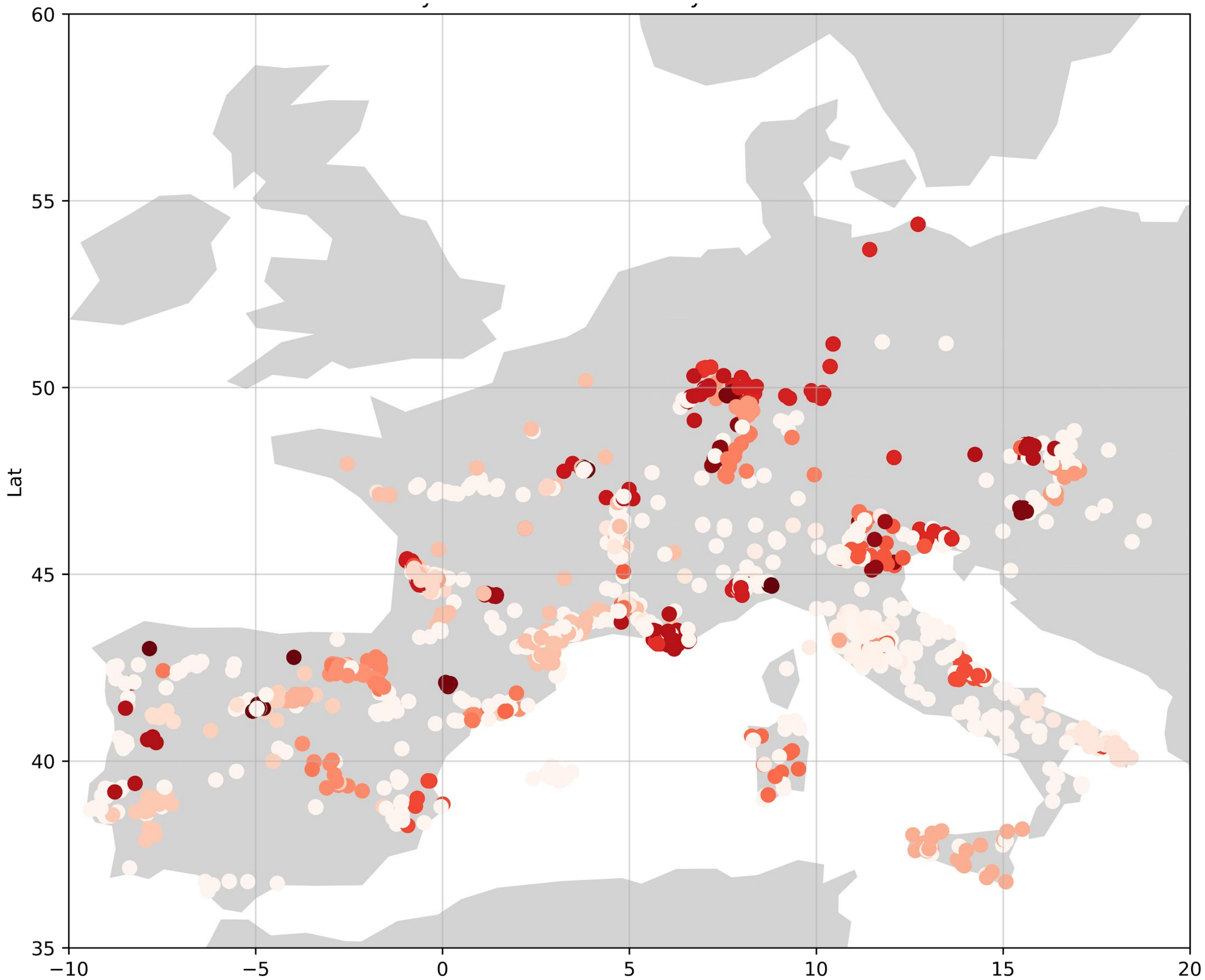
Colored by DBSCAN clusters

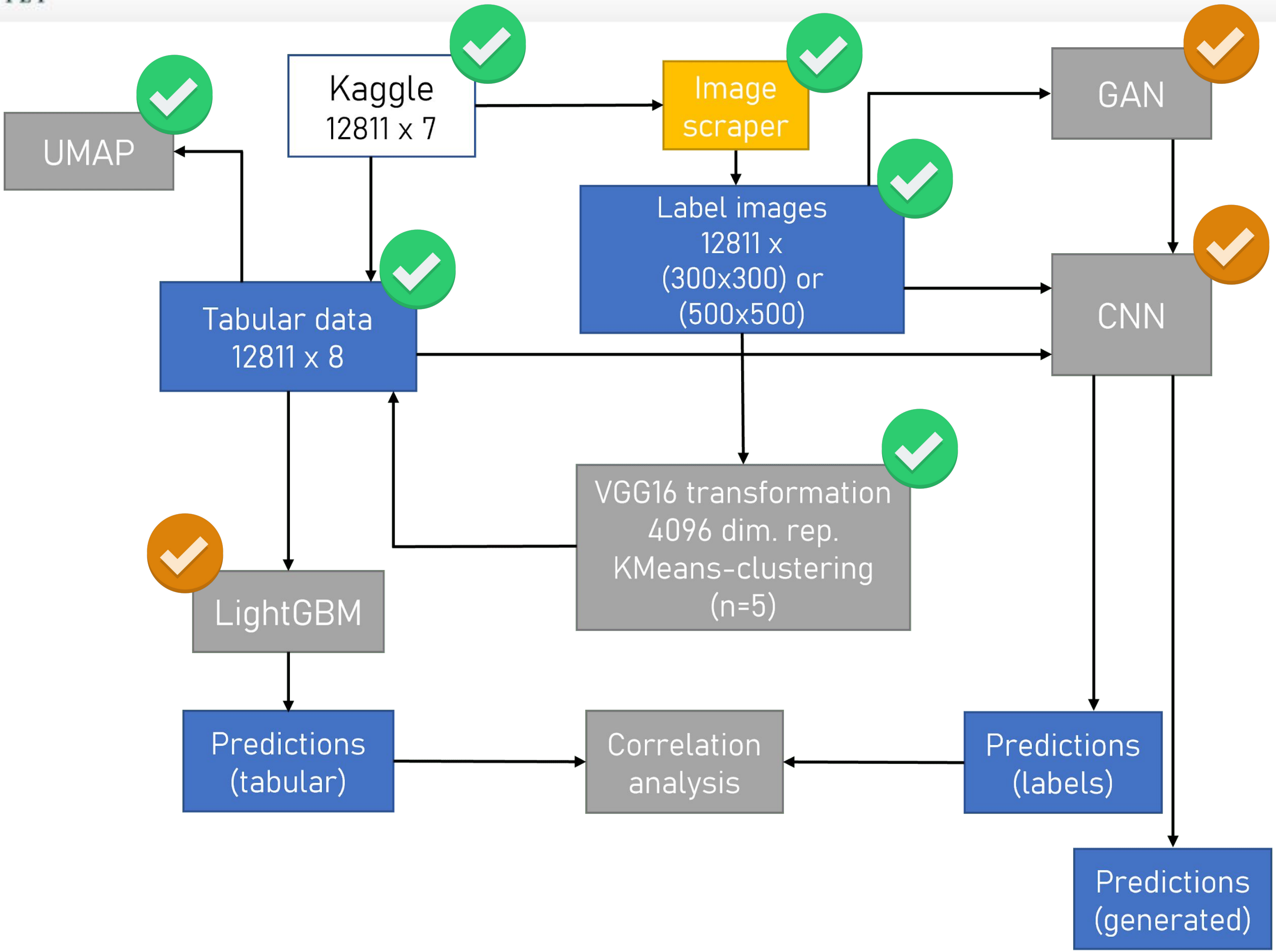


DBSCAN uniformly colors clusters in the two component plane by UMAP

Geopandas: a library for plotting maps.
(Among other things)

Comparable to the UMAP, wines assigned to the same cluster also show up in the same area




Kaggle
12811 x 7
Image scraper
GAN
UMAP
Label images
12811 x
(300x300) or
(500x500)
Tabular data
12811 x 8
CNN
VGG16 transformation
4096 dim. rep.
KMeans-clustering
(n=5)
LightGBM
Predictions
(tabular)
Correlation
analysis
Predictions
(labels)
Predictions
(generated)

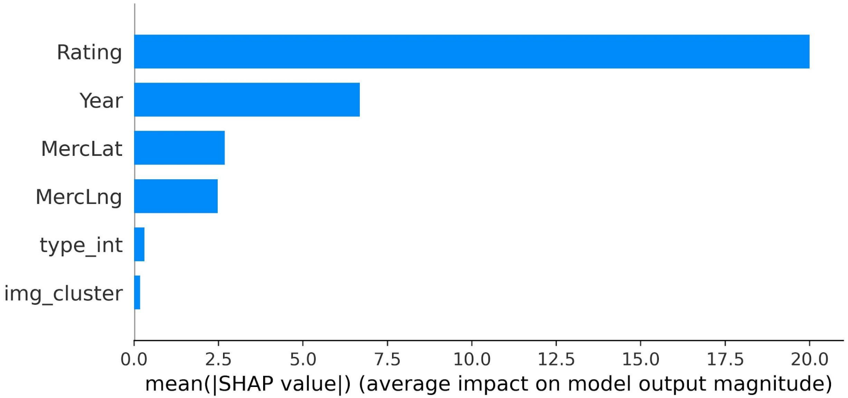
LightGBM

LightGBM

SHAP

Optuna

Price regression



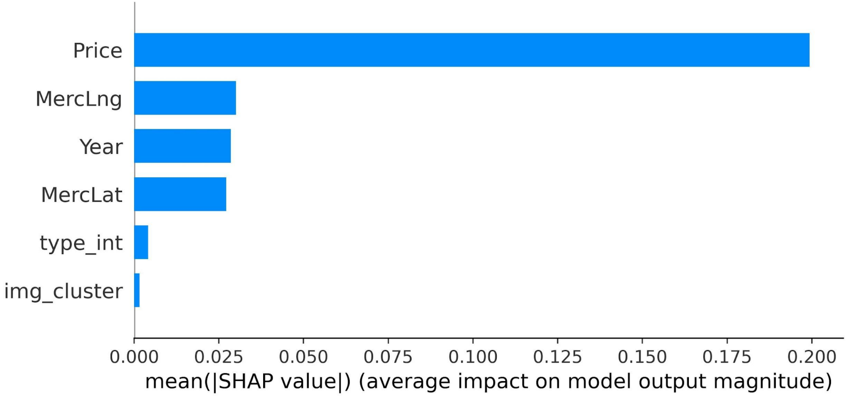

```
Number of finished trials: 350
Best trial:
  Value: 0.17479515859838068
  Params:
    objective: regression
    metric: l2
    boosting_type: gbdt
    lambda_l1: 9.951685044230247e-06
    lambda_l2: 3.0056626435477843e-05
    num_leaves: 32
    bagging_fraction: 0.8565231072881204
    bagging_freq: 1
    min_child_samples: 12
```

350 trials

Metric (LGBM): L2

Metric (Optuna): RMSE

Rating regression

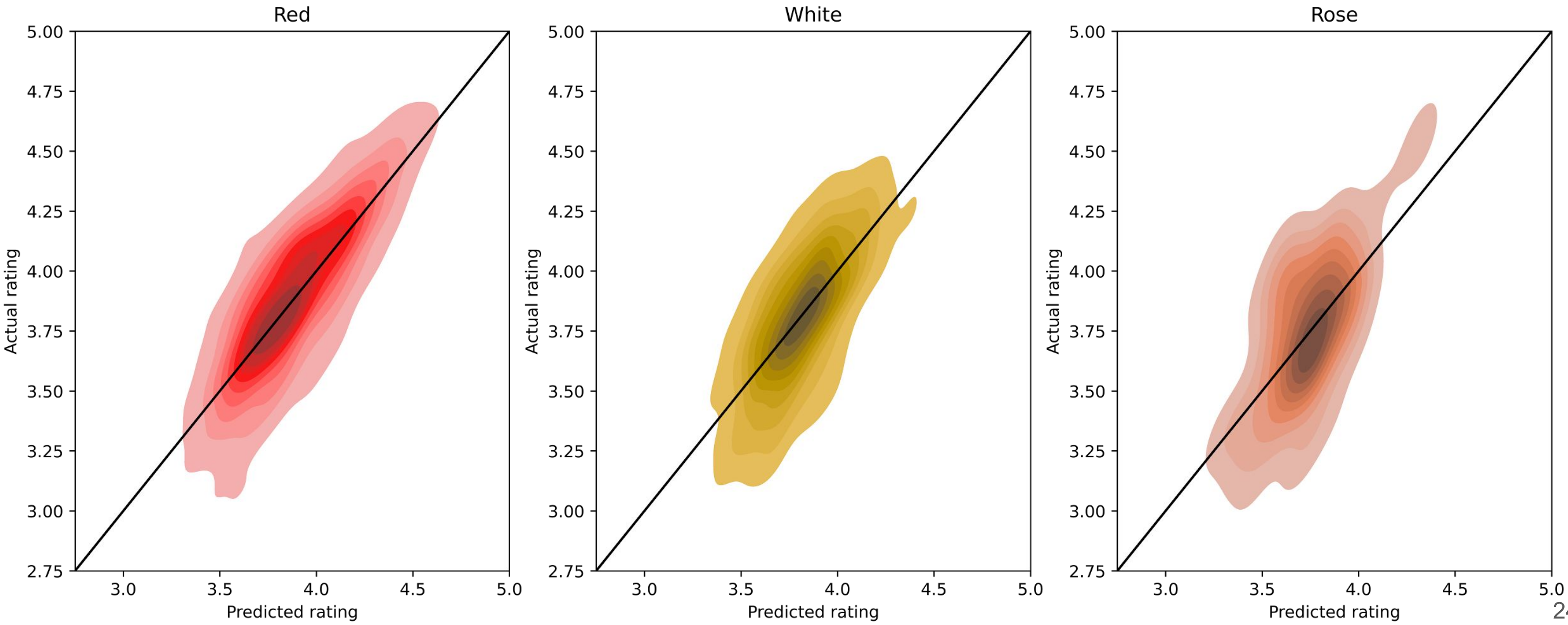


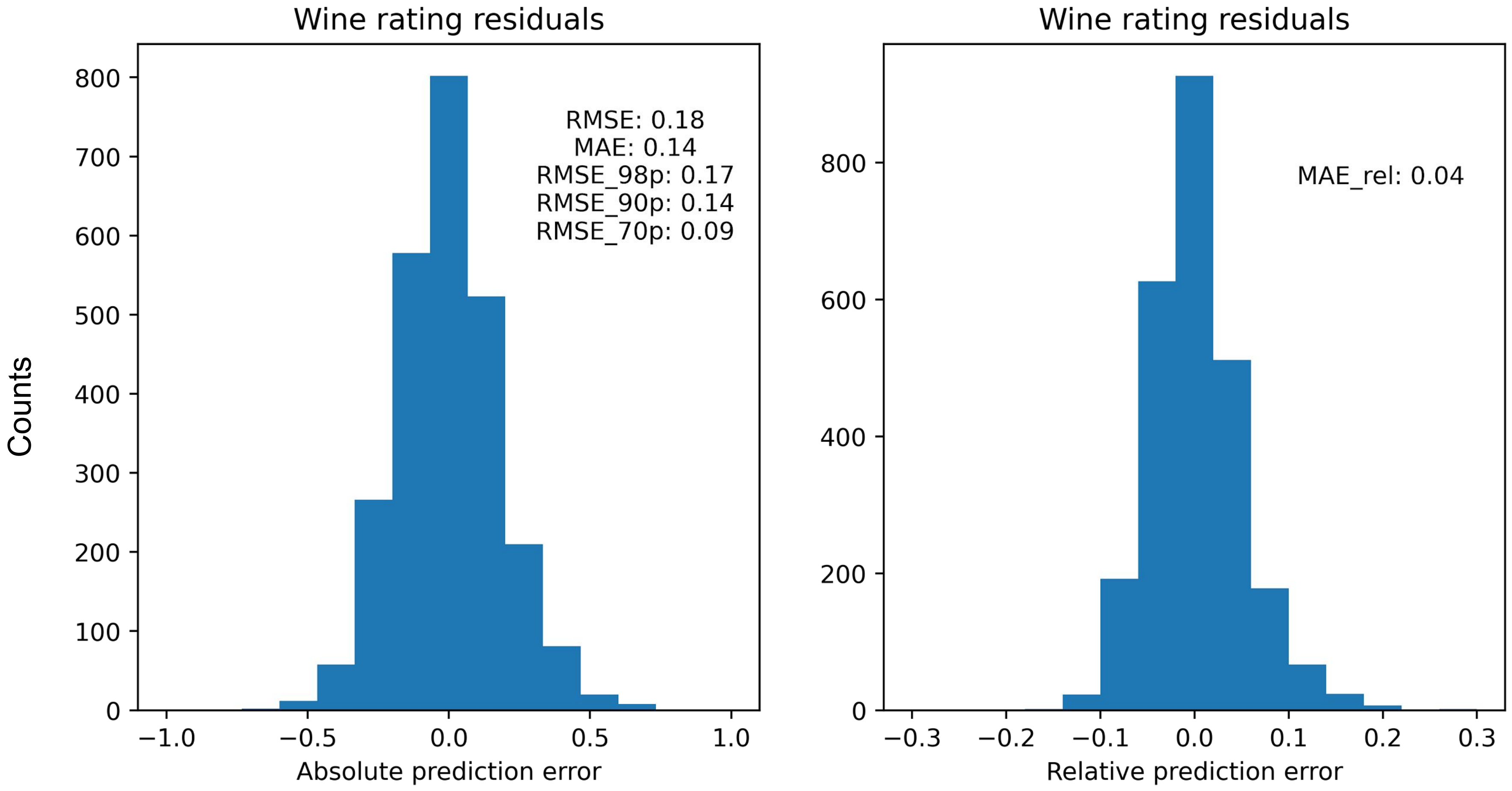

```
Number of finished trials: 350
Best trial:
  Value: 30.962572509965124
  Params:
    objective: regression
    metric: l2
    boosting_type: gbdt
    lambda_l1: 9.700136027458894
    lambda_l2: 0.4129823906468111
    num_leaves: 194
    bagging_fraction: 0.8681925817491165
    bagging_freq: 7
    min_child_samples: 21
```

LightGBM regressor

Variables: Type, price, year, MercLat, MercLng, img_cluster

RMSE 0.18, 12811 entries

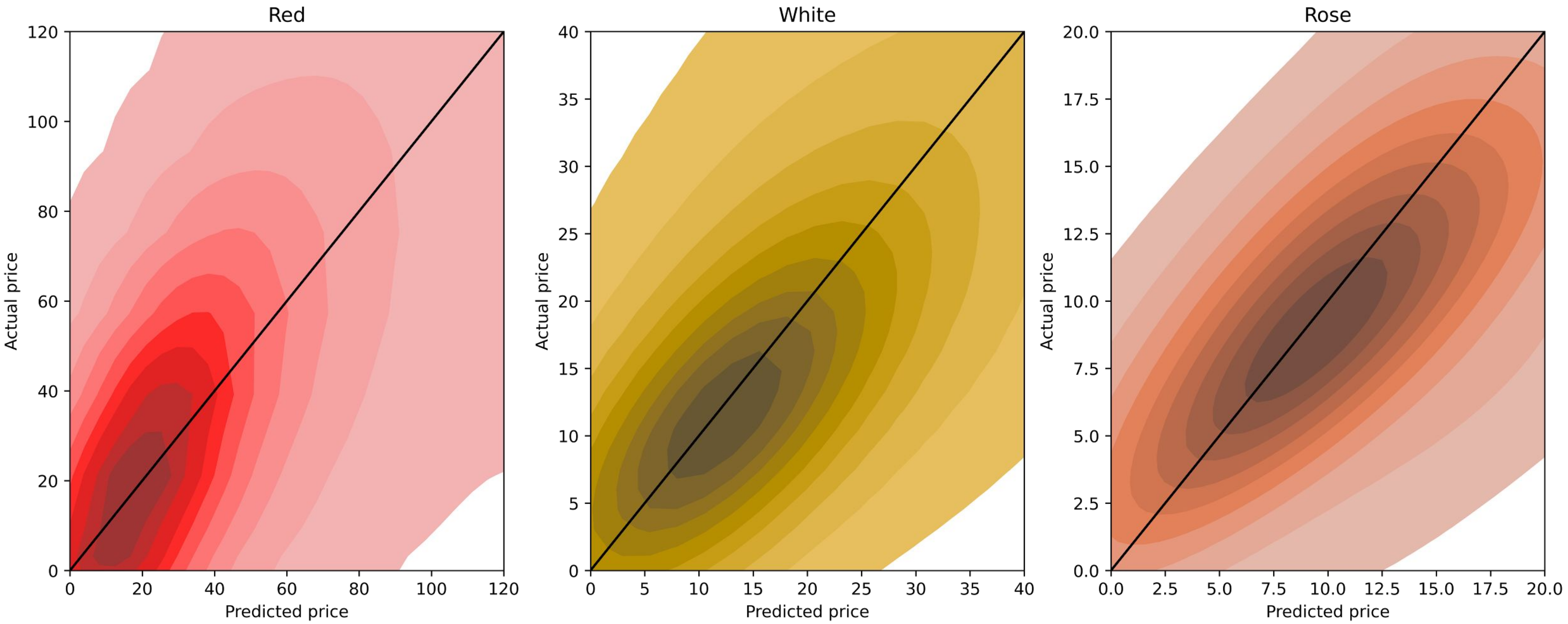


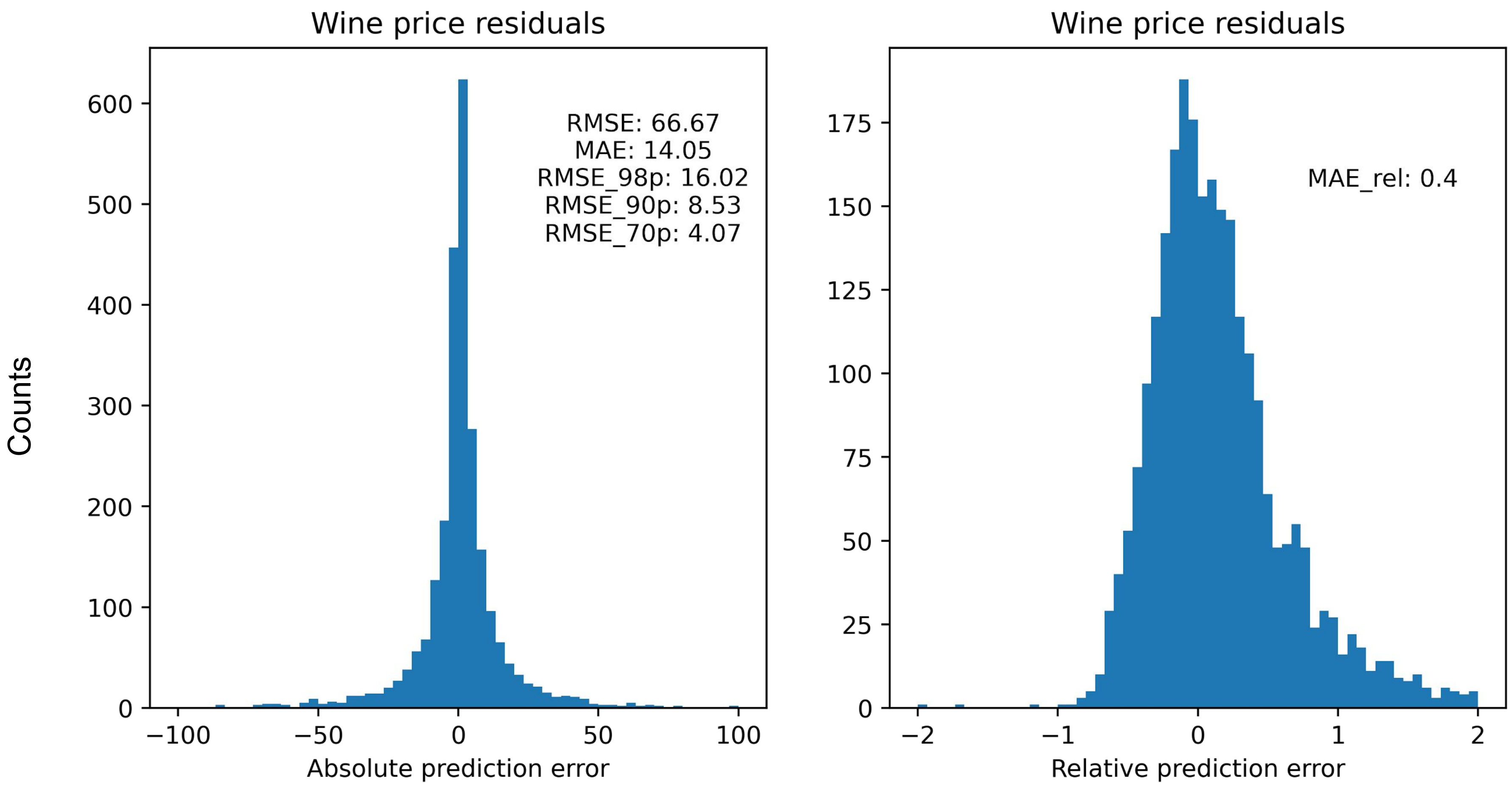

Counts
Wine rating residuals
RMSE: 0.18
MAE: 0.14
RMSE_98p: 0.17
RMSE_90p: 0.14
RMSE_70p: 0.09
0
100
200
300
400
500
600
700
800
−1.0
−0.5
0.0
0.5
1.0
Absolute prediction error
Wine rating residuals
MAE_rel: 0.04
0
200
400
600
800
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
Relative prediction error

LightGBM regressor

Variables: Type, rating, year, MercLat, MercLng, img_cluster

RMSE 66.67, 12811 entries




Counts
Wine price residuals
RMSE: 66.67
MAE: 14.05
RMSE_98p: 16.02
RMSE_90p: 8.53
RMSE_70p: 4.07
Absolute prediction error
Wine price residuals
MAE_rel: 0.4
Relative prediction error

A Generative Adversarial Network

(And Variational Auto Encoder)

Variational Auto Encoder

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 128, 128]	896
BatchNorm2d-2	[-1, 32, 128, 128]	64
LeakyReLU-3	[-1, 32, 128, 128]	0
Conv2d-4	[-1, 64, 64, 64]	18,496
BatchNorm2d-5	[-1, 64, 64, 64]	128
LeakyReLU-6	[-1, 64, 64, 64]	0
Conv2d-7	[-1, 128, 32, 32]	73,856
BatchNorm2d-8	[-1, 128, 32, 32]	256
LeakyReLU-9	[-1, 128, 32, 32]	0
Conv2d-10	[-1, 256, 16, 16]	295,168
BatchNorm2d-11	[-1, 256, 16, 16]	512
LeakyReLU-12	[-1, 256, 16, 16]	0
Conv2d-13	[-1, 512, 8, 8]	1,180,160
BatchNorm2d-14	[-1, 512, 8, 8]	1,024
LeakyReLU-15	[-1, 512, 8, 8]	0
Linear-16	[-1, 128]	4,194,432
Linear-17	[-1, 128]	4,194,432
Linear-18	[-1, 32768]	4,227,072
ConvTranspose2d-19	[-1, 256, 16, 16]	1,179,904
BatchNorm2d-20	[-1, 256, 16, 16]	512
LeakyReLU-21	[-1, 256, 16, 16]	0
ConvTranspose2d-22	[-1, 128, 32, 32]	295,040
BatchNorm2d-23	[-1, 128, 32, 32]	256
LeakyReLU-24	[-1, 128, 32, 32]	0
ConvTranspose2d-25	[-1, 64, 64, 64]	73,792
BatchNorm2d-26	[-1, 64, 64, 64]	128
LeakyReLU-27	[-1, 64, 64, 64]	0
ConvTranspose2d-28	[-1, 32, 128, 128]	18,464
BatchNorm2d-29	[-1, 32, 128, 128]	64
LeakyReLU-30	[-1, 32, 128, 128]	0
ConvTranspose2d-31	[-1, 32, 256, 256]	9,248
BatchNorm2d-32	[-1, 32, 256, 256]	64
LeakyReLU-33	[-1, 32, 256, 256]	0
Conv2d-34	[-1, 3, 256, 256]	867
Tanh-35	[-1, 3, 256, 256]	0

Hidden layers: [32,64,128,256,512]

Latent dim: 128

lr: 0.001

Activation function: LeakyReLU

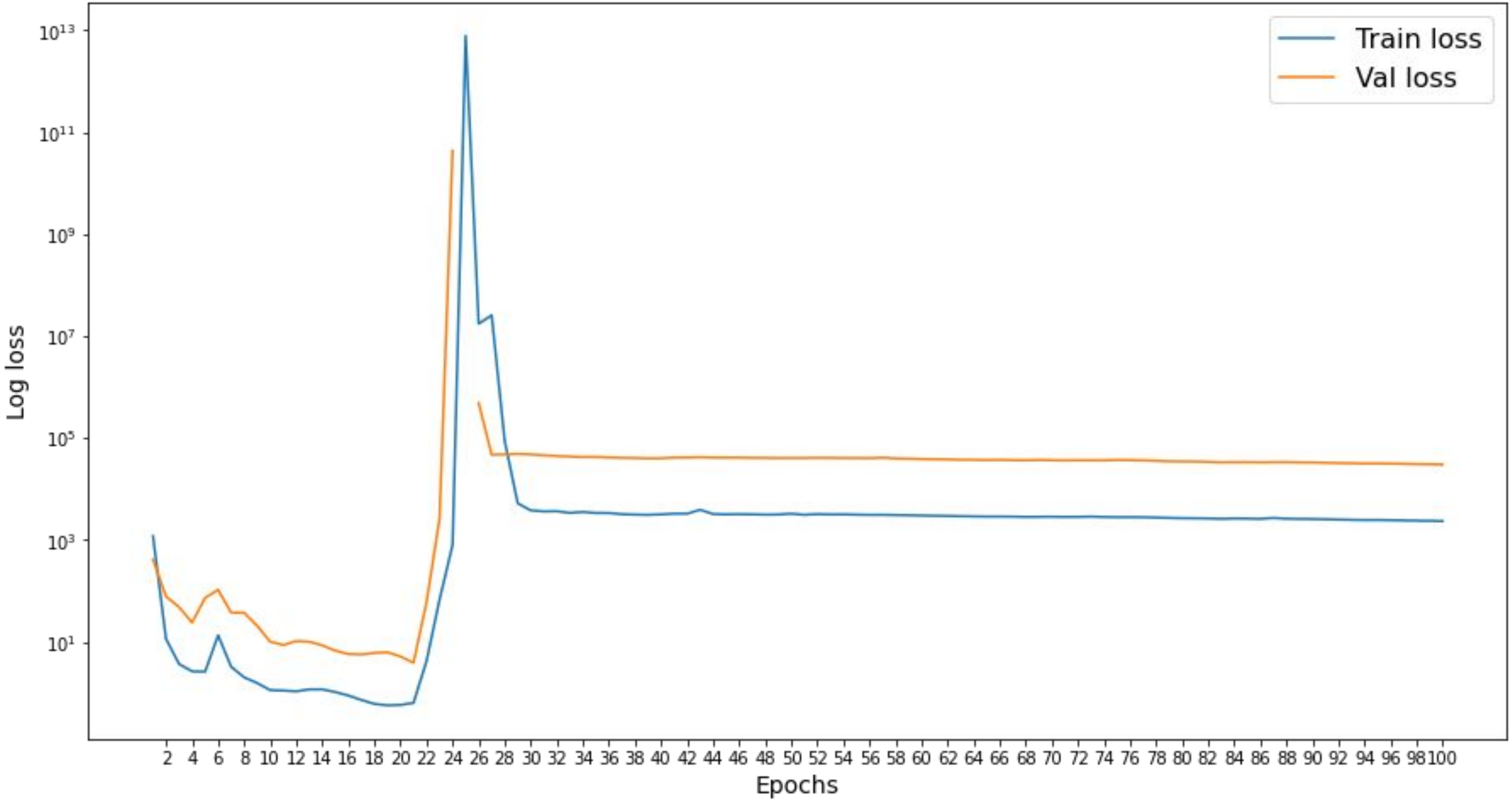
Pooling : BatchNorm2d

End activation: Tanh

Loss: KL_loss * 0.5 + MSE

```
----------------------------------
Total params: 15,764,835
Trainable params: 15,764,835
Non-trainable params: 0
----------------------------------
```

VAE loss by epoch



Average prof red wine

VAE generated

… Perhaps a different approach?

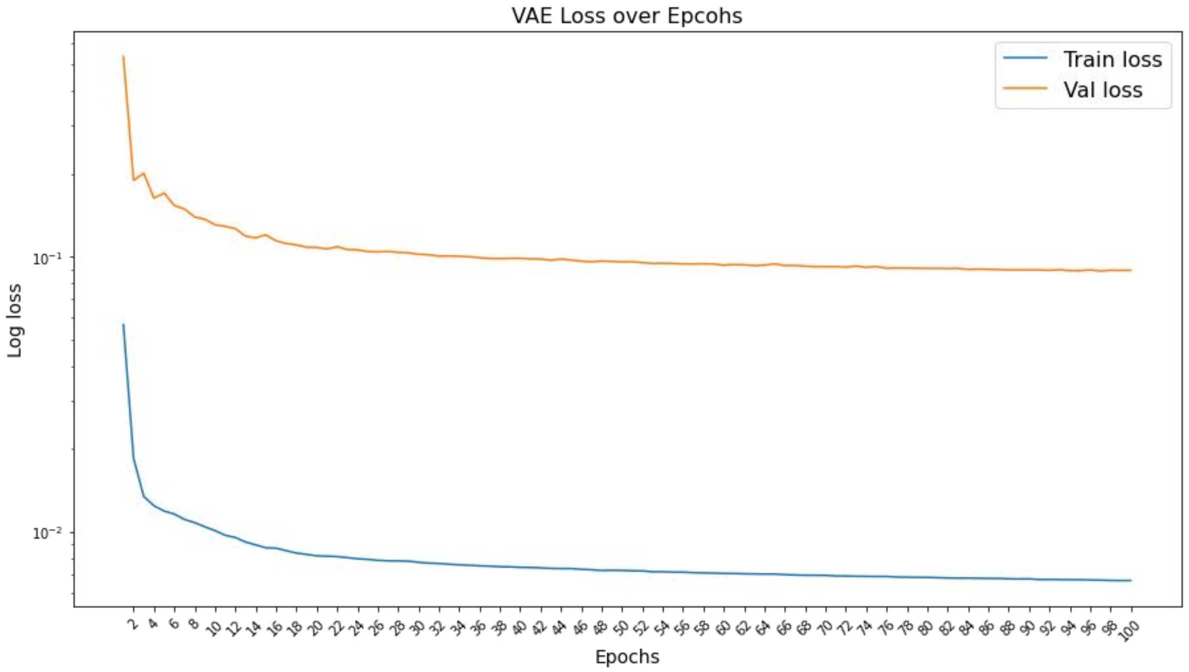
Hidden layers: [8,16,32,64]

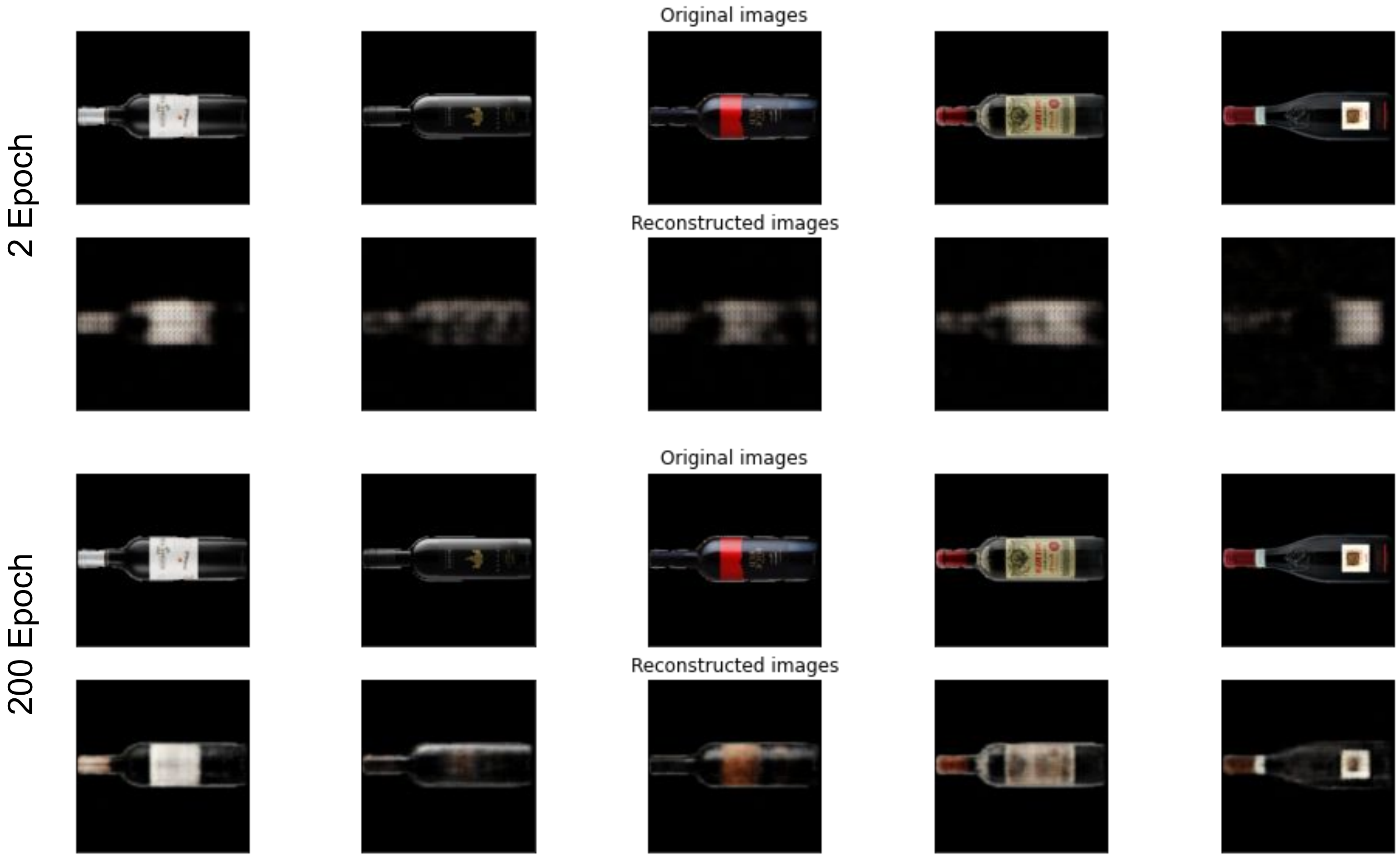
Latent dim: 128

lr: 0.001

End activation: abs(Tanh)

Loss: KL_loss * 0.001 + MSE




2 Epoch
Original images
Reconstructed images
200 Epoch
Original images
Reconstructed images

Generative Adversarial Network

Generator

```
Laten dim: 300

Linear Layer: 640000

(Tran)Conv2d layer: [256,64,64,64,64]

Activation fn: LeakyRelu

Lr: [0.0002]

Loss: Binary Cross entropy

Output shape: [300,300,3]

------------------------------------
Total params: 192,740,227
Trainable params: 192,740,227
Non-trainable params: 0
------------------------------------
```

Discriminator

```
Conv2d layer: [64,128,128,256]

Activation fn: LeakyRelu

Linear layer: [640000]

Pooling: Dropout

Lr: [0.0002]

Loss: Binary Cross entropy

Output shape: [1]

------------------------------------
Total params: 870,721
Trainable params: 870,721
Non-trainable params: 0
------------------------------------
```

Epoch 350

A Convolutional Neural Network

for predicting rating and price

CNN Output

Three different cases

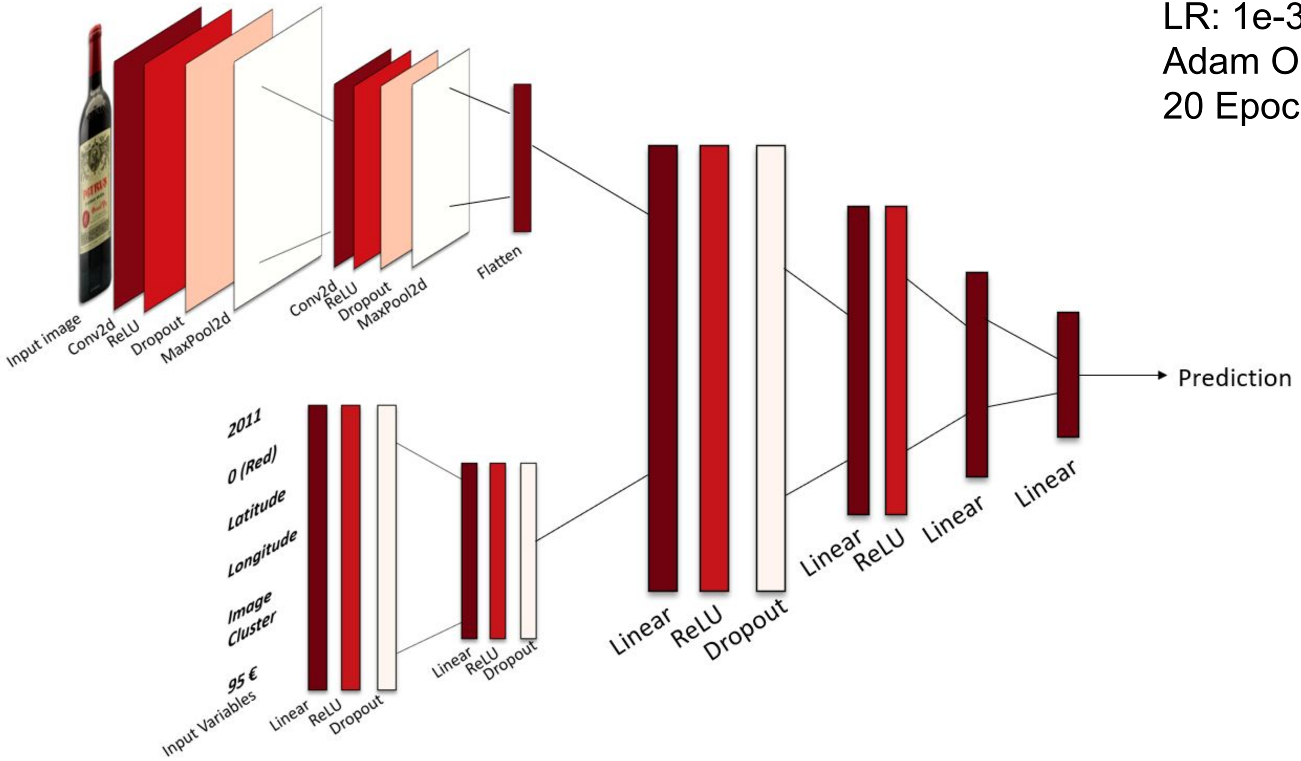
- Regression based on image

- Regression based on image + variables (same variables as in LGBM)

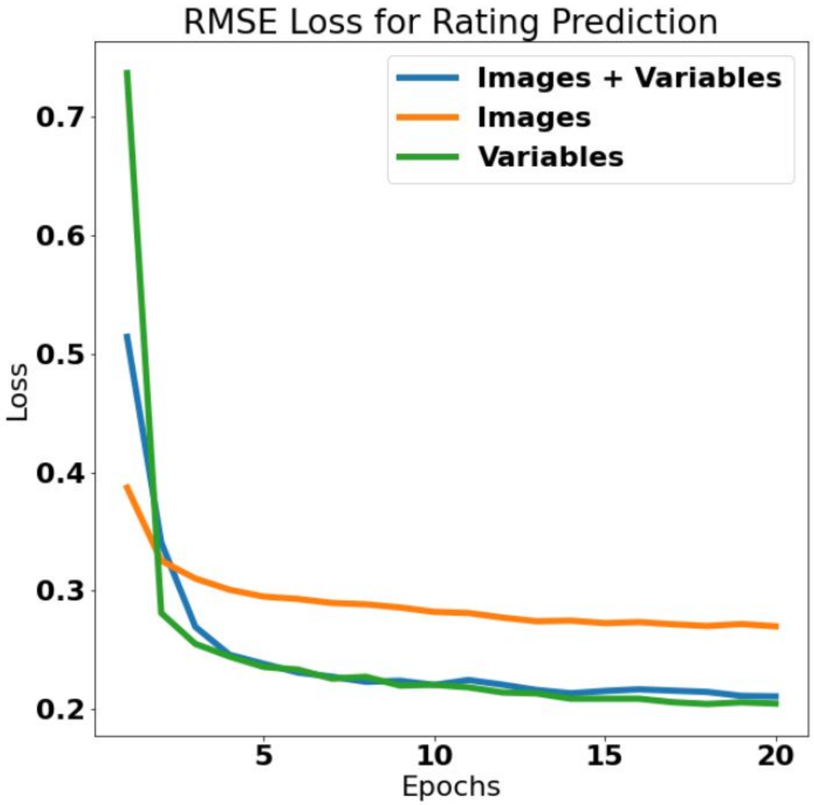
- Regression based on variables

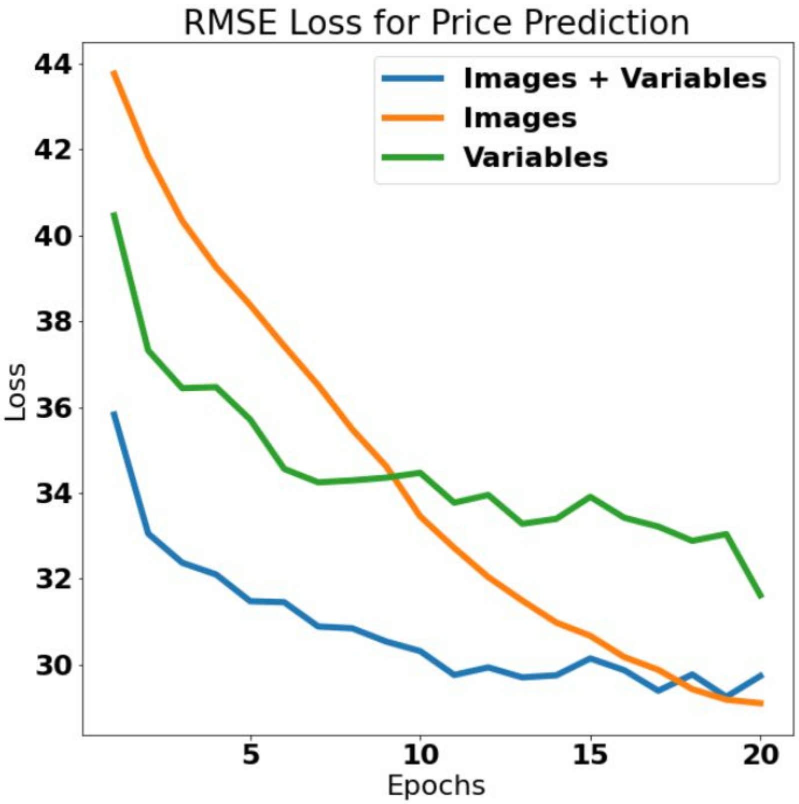
CNN structure - images and variables

LR: 1e-3
Adam Optimizer
20 Epochs

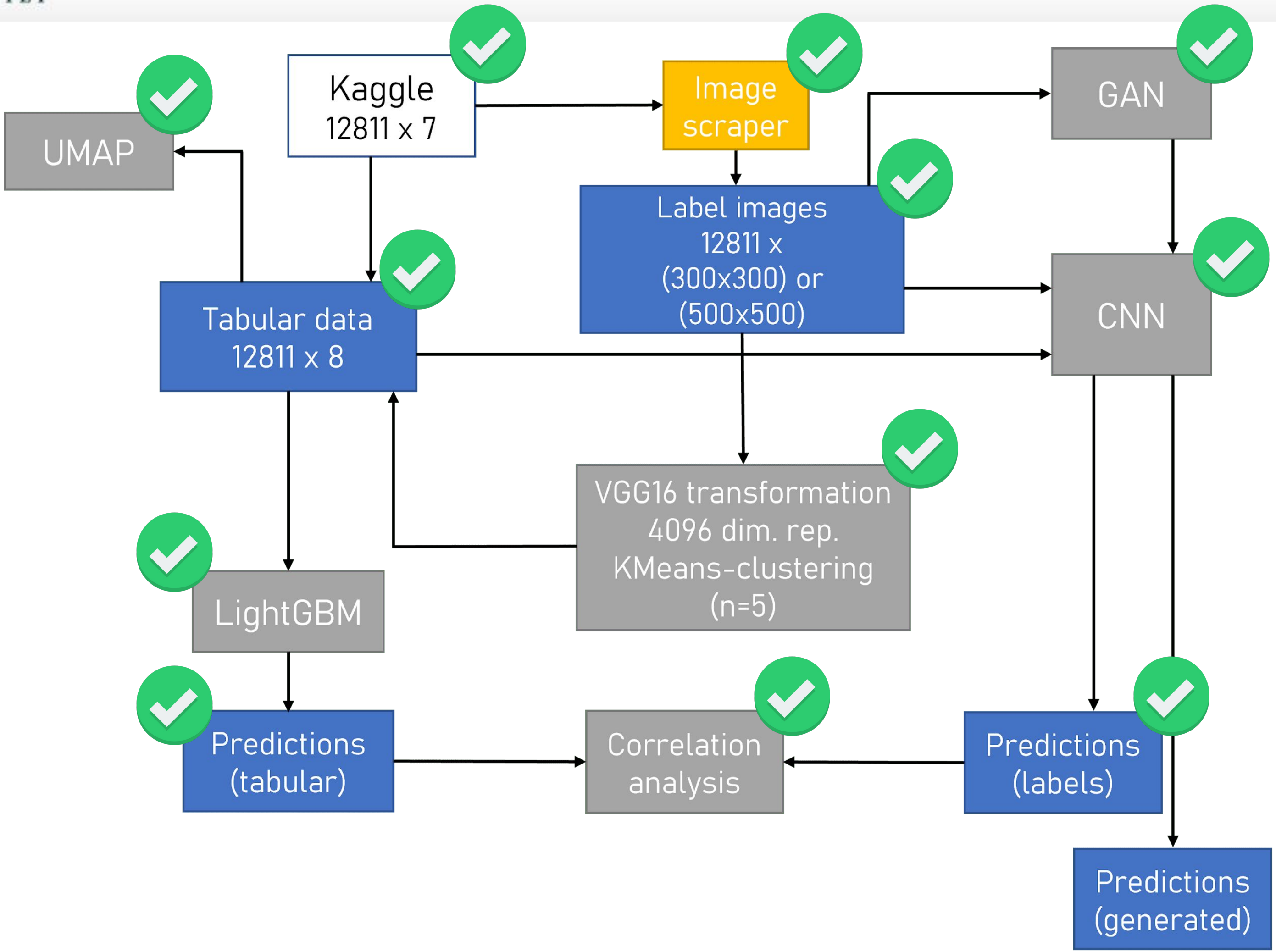


RMSE (training) loss - Rating and Price

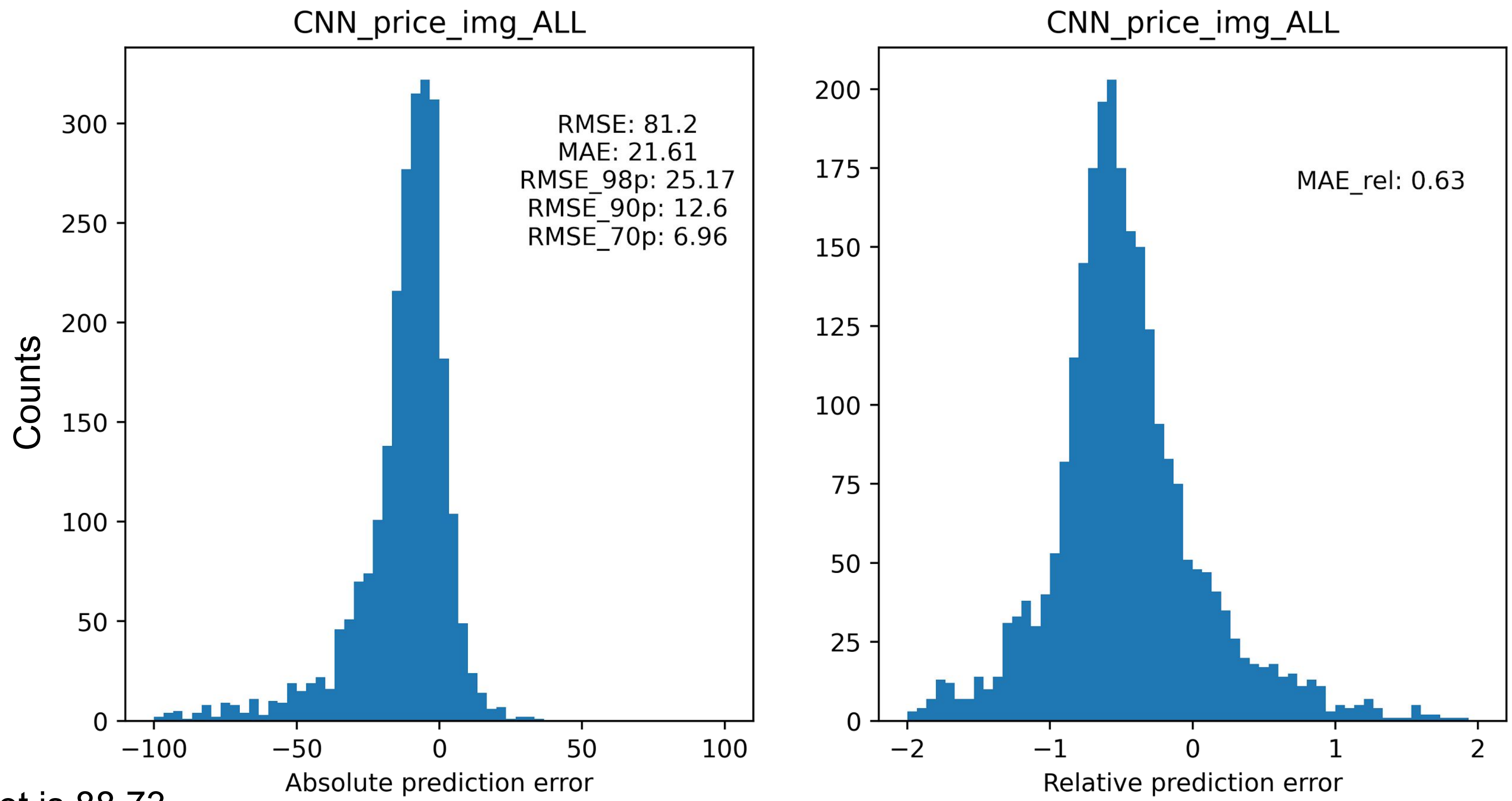




Learns specific kernel for outliers?

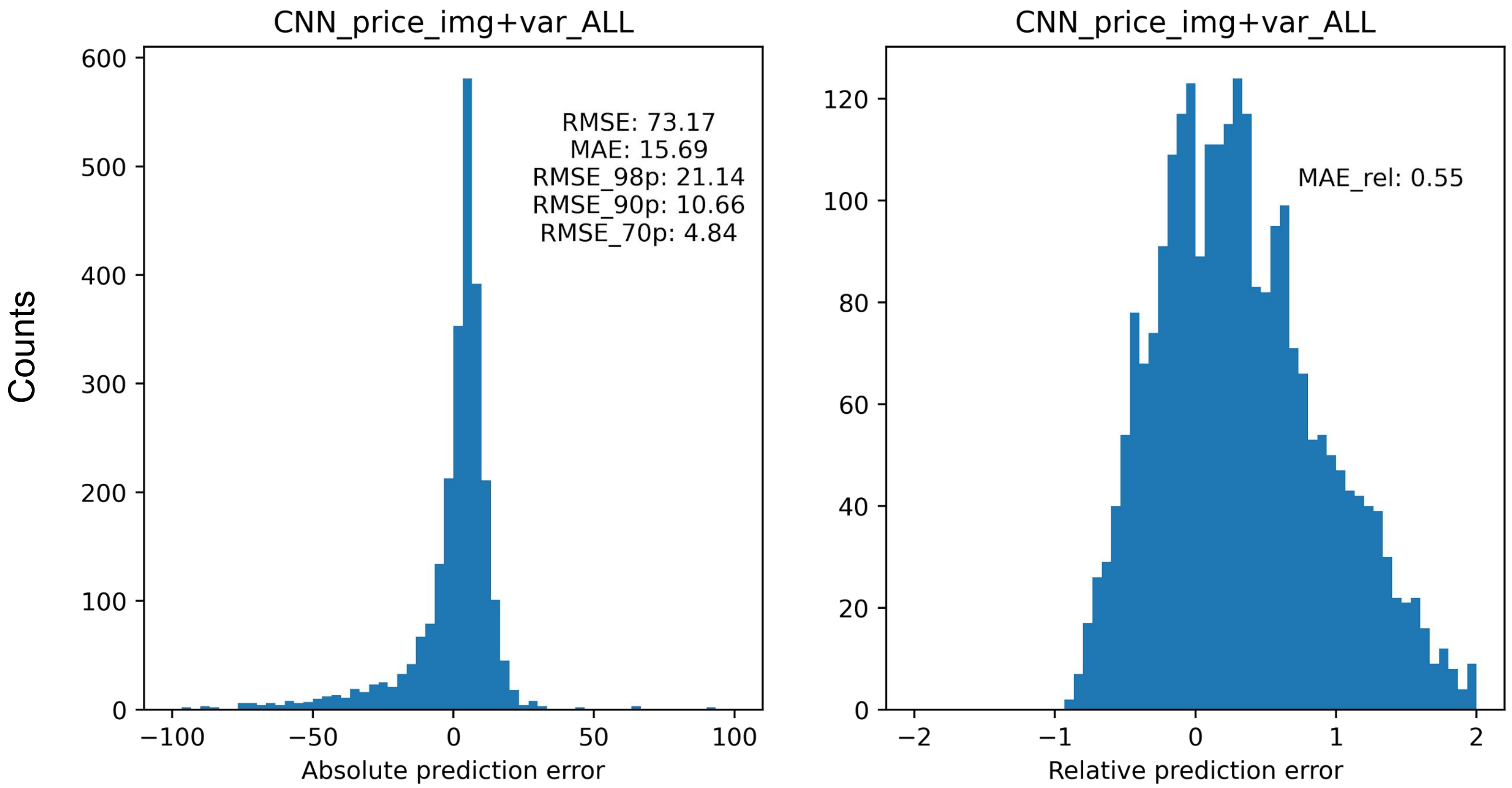

Kaggle
12811 x 7
Image scraper
GAN
UMAP
Label images
12811 x
(300x300) or
(500x500)
Tabular data
12811 x 8
CNN
VGG16 transformation
4096 dim. rep.
KMeans-clustering
(n=5)
LightGBM
Predictions
(tabular)
Correlation
analysis
Predictions
(labels)
Predictions
(generated)

CNN regression residuals (predicting by label image)

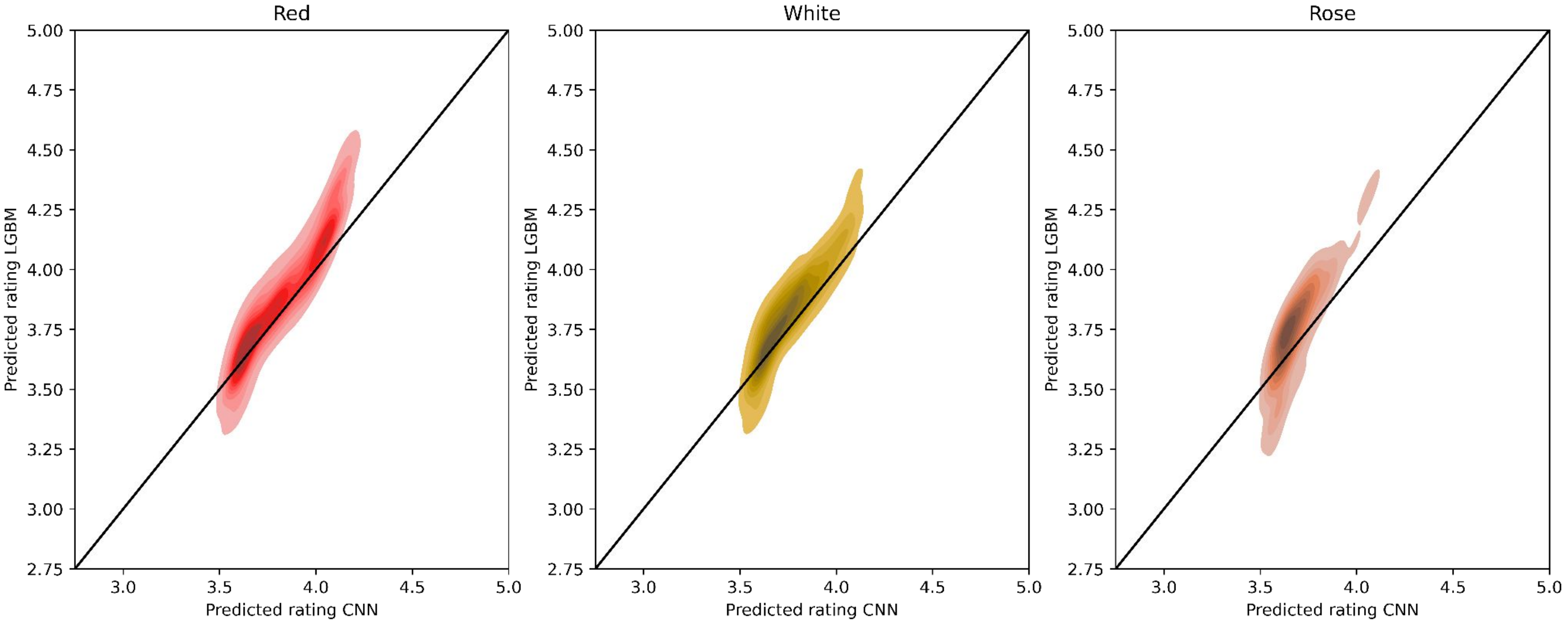


std of test set is 88.73

CNN regression residuals (predicting by label image + variables)



LGBM vs. CNN (Images and Variables) - Ratings



Conclusion

Rating:
The best CNN-based model (vars), RMSE: 0.20
LGBM, RMSE: 0.18

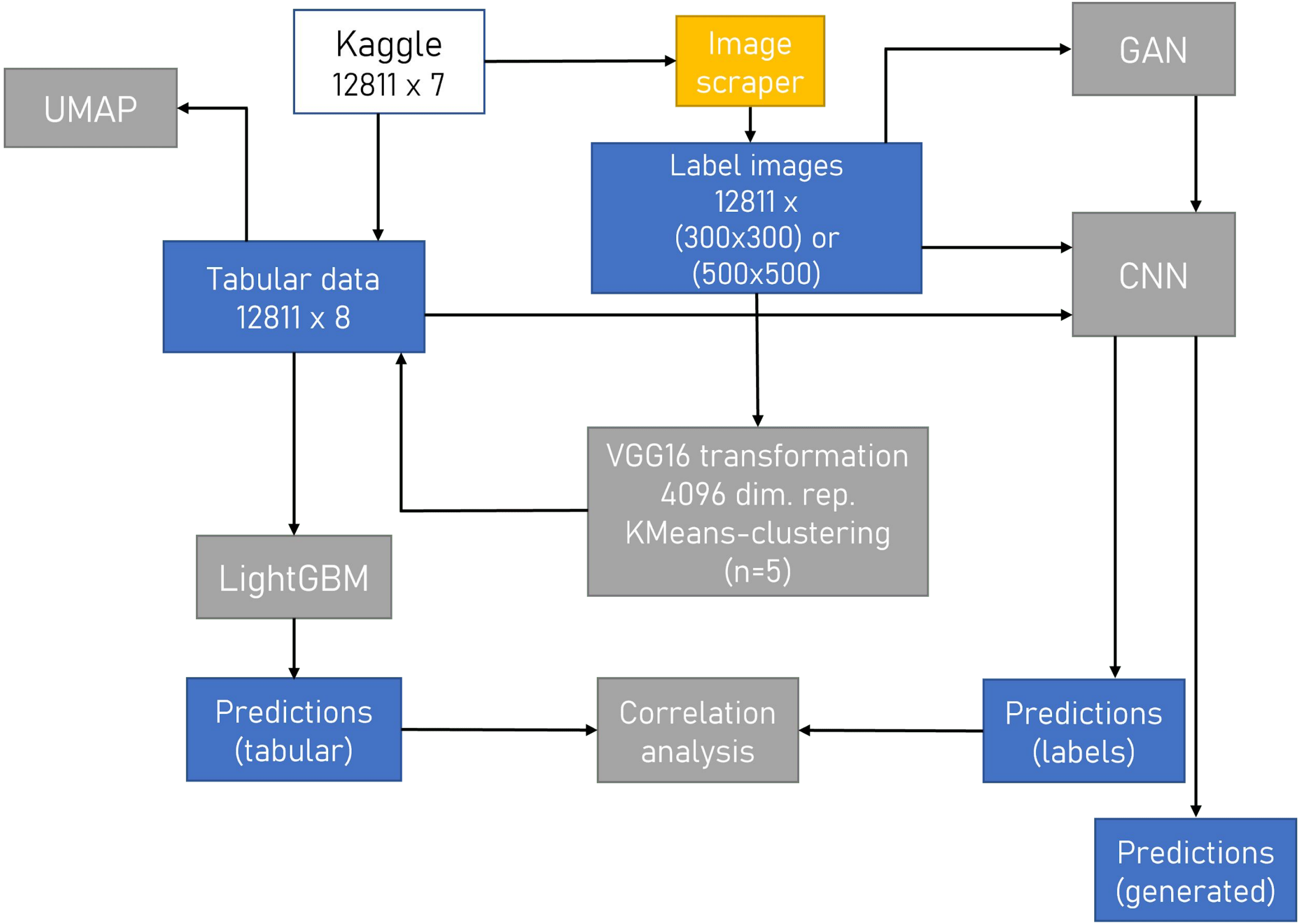
Price:
The best CNN-based model (vars), RMSE: 72.32
LGBM, RMSE: 66.67

Image-based CNN does learn during training, however we expect an ensemble model including both a tree-based and CNN is the way to go to incorporate label images.

We do manage to use ML methods to predict ratings and price, though there was no information to gain from the wine label images.

Further Work

- Scrap higher resolution pictures
- Addition of one-hot encoded categorical variables for CNN
- Utilize BOW from wine names
- SHAP on CNN
- Given working VAE
  - Train Linear NN to produce latent space of given picture from corresponding meta-data
  - Produce never seen before wine bottles through decoder given custom meta-data input
- Further improve GAN
  - Estimate rating and/or price from generated picture


Kaggle
12811 x 7
Image scraper
GAN
UMAP
Label images
12811 x
(300x300) or
(500x500)
Tabular data
12811 x 8
CNN
VGG16 transformation
4096 dim. rep.
KMeans-clustering
(n=5)
LightGBM
Predictions
(tabular)
Correlation
analysis
Predictions
(labels)
Predictions
(generated)

Questions?

APPENDIX

$$\underbrace{\text{Red} \quad \text{White} \quad \text{Rose}}_{\text{One-out-of-k encoding}}$$

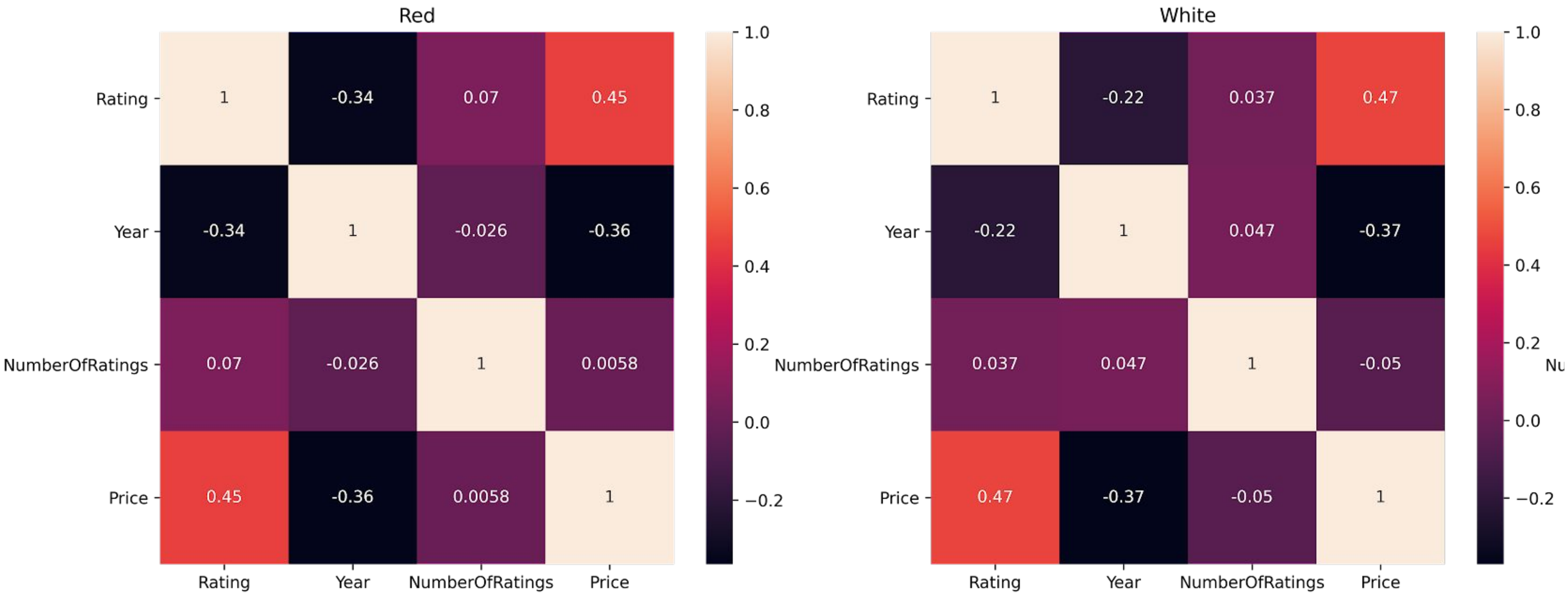
	Rating	NumberOfRatings	Price	Year	Red	White	Rose
count	12811.000000	12811.000000	12811.000000	12811.000000	12811.000000	12811.000000	12811.000000
mean	3.864601	331.683163	32.902042	2015.607681	0.675825	0.293420	0.030755
std	0.297821	734.647832	72.460263	3.103701	0.468084	0.455347	0.172659
min	2.500000	25.000000	3.550000	1988.000000	0.000000	0.000000	0.000000
25%	3.700000	55.000000	9.900000	2015.000000	0.000000	0.000000	0.000000
50%	3.900000	122.000000	15.900000	2016.000000	1.000000	0.000000	0.000000
75%	4.100000	312.000000	31.850000	2018.000000	1.000000	1.000000	0.000000
max	4.900000	20293.000000	3410.790000	2020.000000	1.000000	1.000000	1.000000

The raw numericals

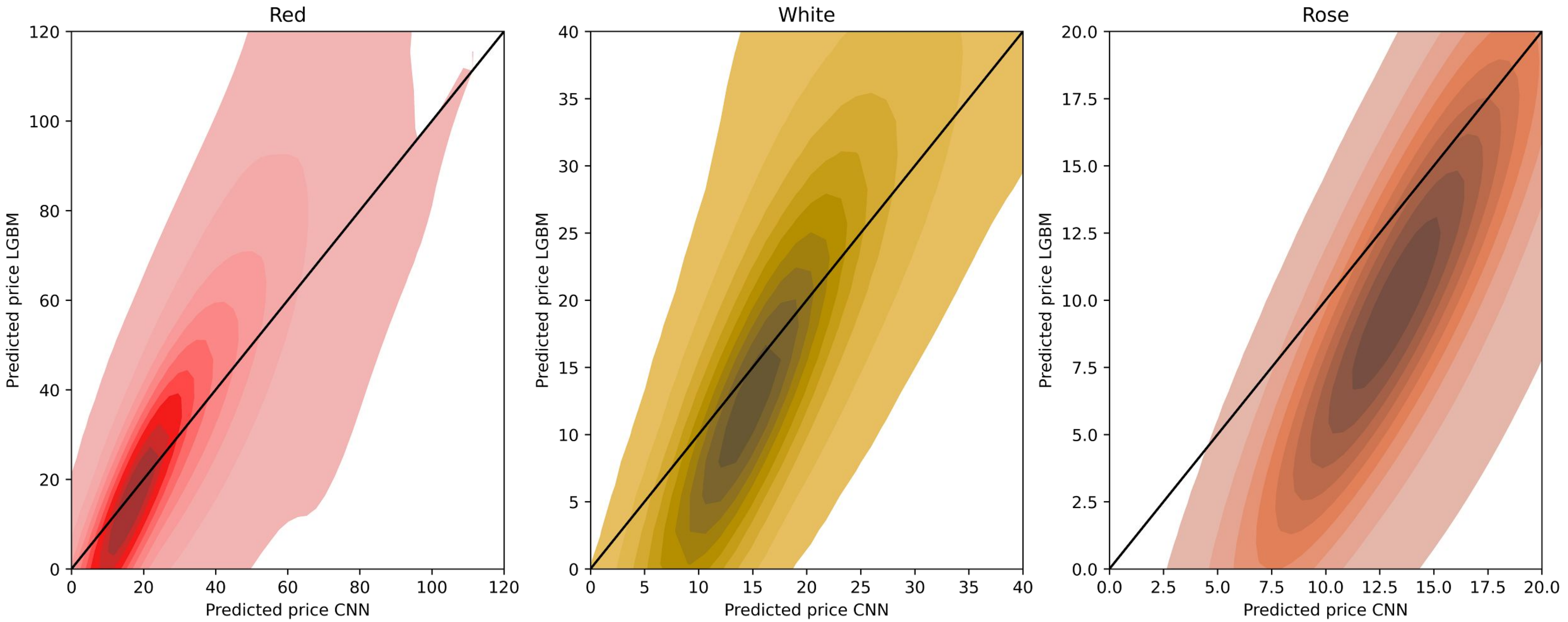
N_unique values of the categorical data

```
Name        10079
Country        32
Region        813
Winery       3264
dtype: int64
```

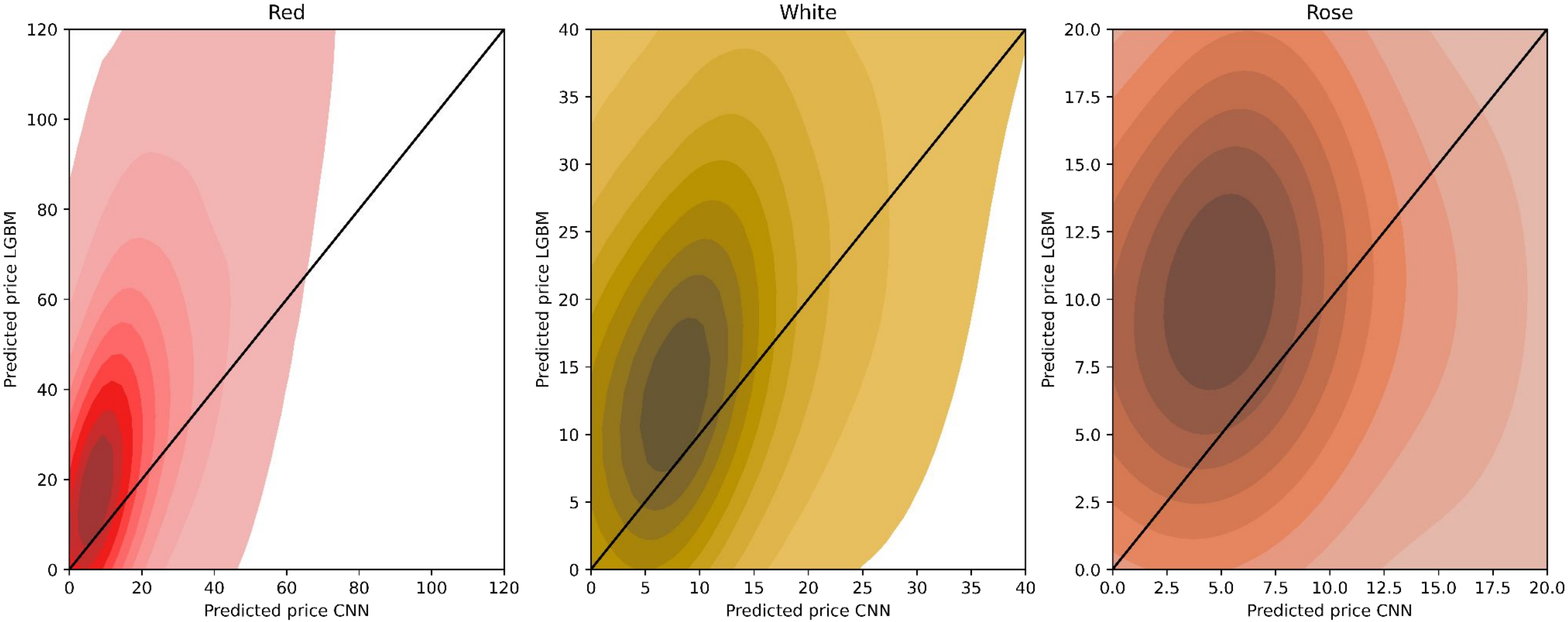
Correlation matrices of red and white wine



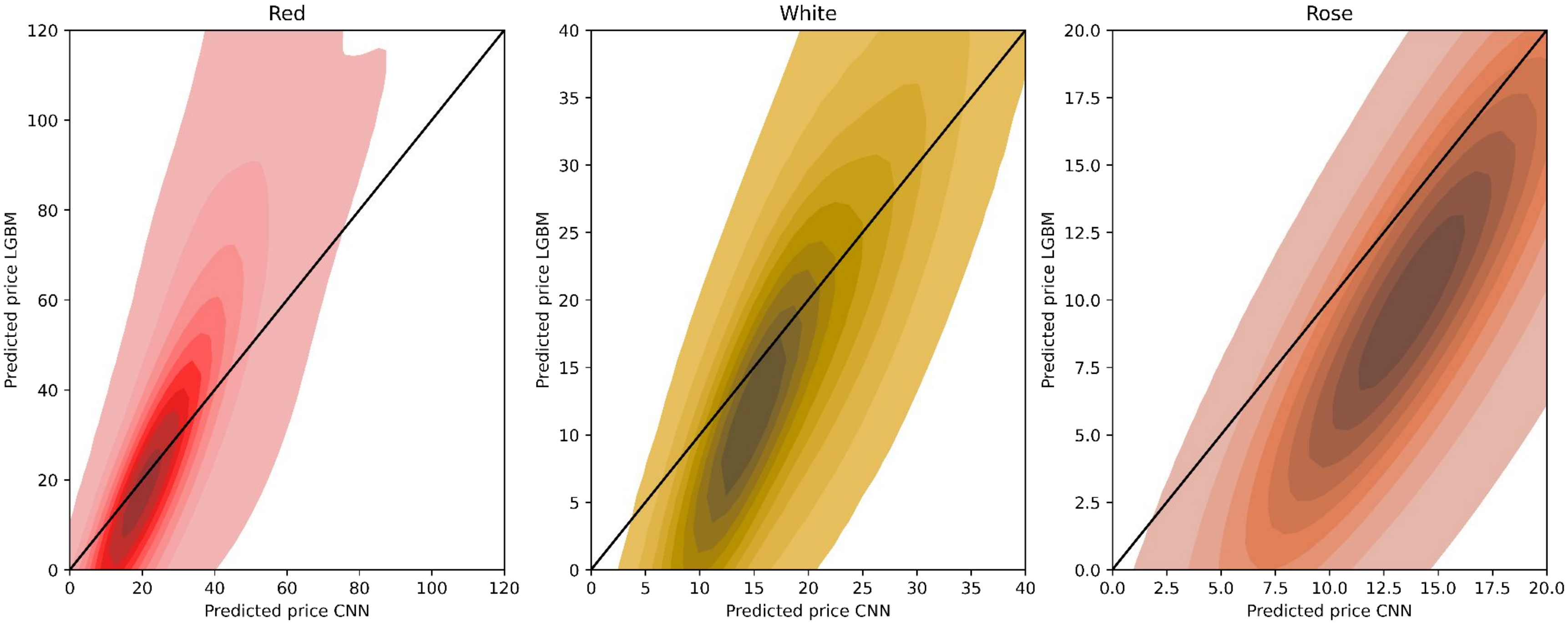
LGBM vs. CNN (variables) - Price Prediction



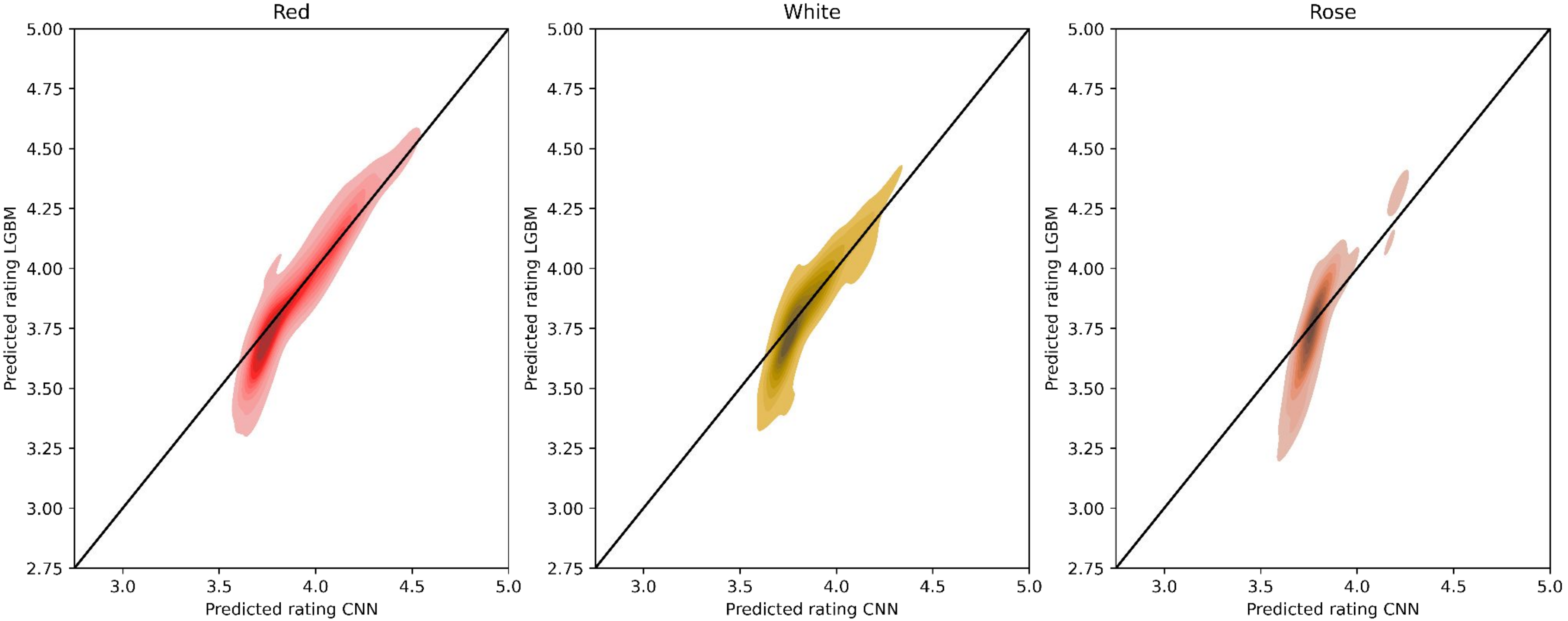
LGBM vs. CNN - Price Prediction - Images



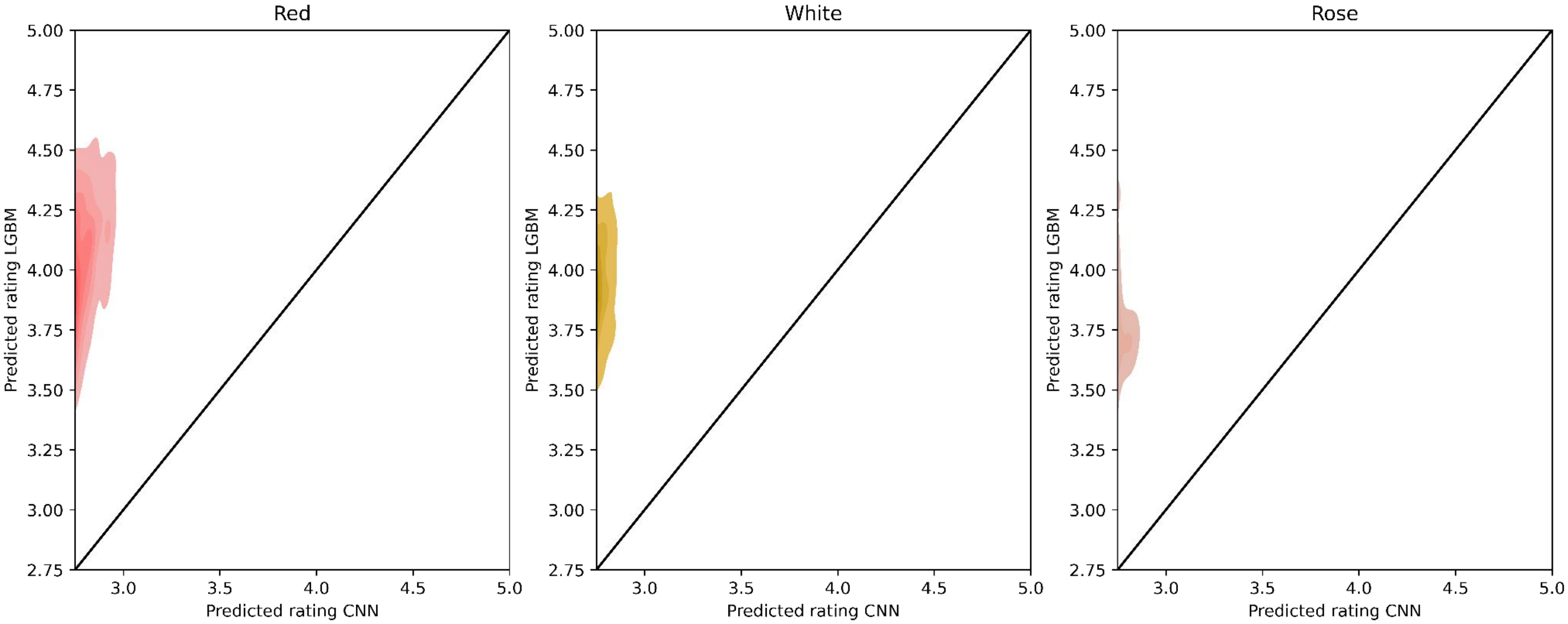
LGBM vs. CNN (Images and Variables)- Price Prediction



LGBM vs. CNN (Variables) - Rating Prediction



LGBM vs. CNN (Images)- Rating Prediction



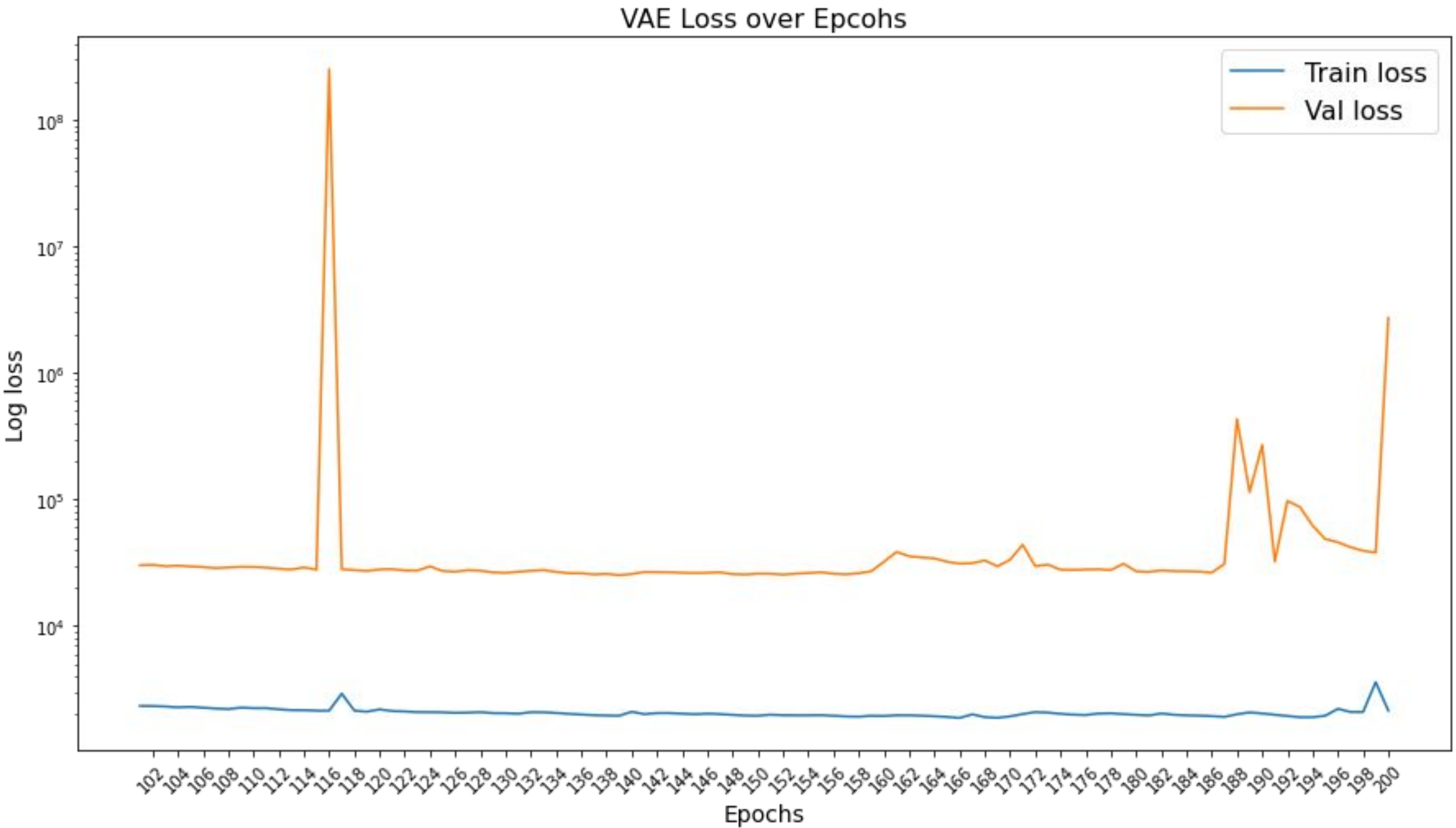
Generative Adversarial Network

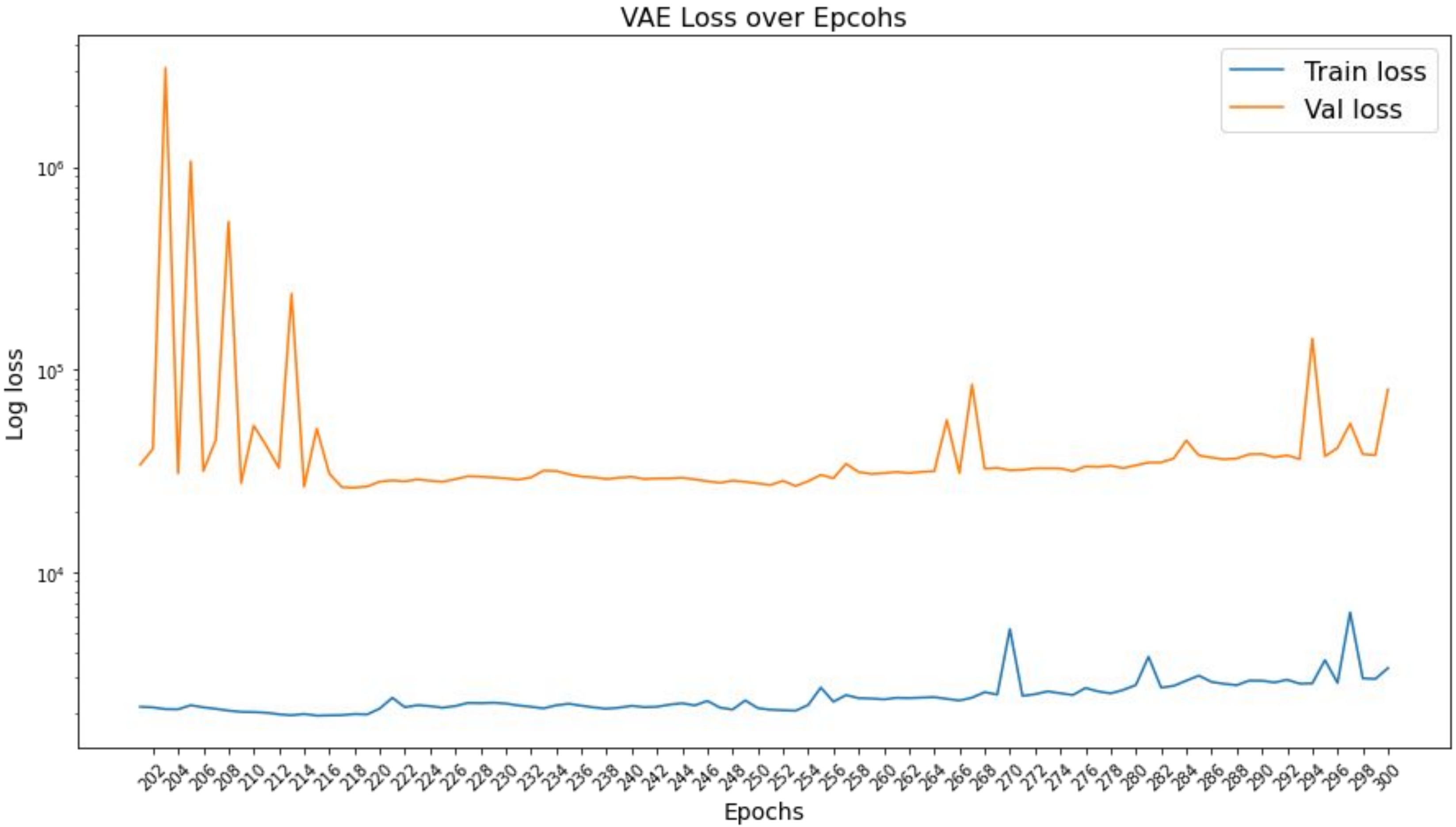
Generator

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_1 (Dense)             (None, 640000)            192640000

 leaky_re_lu_4 (LeakyReLU)   (None, 640000)            0

 reshape (Reshape)           (None, 50, 50, 256)       0

 conv2d_transpose (Conv2DTra  (None, 50, 50, 64)       65600
 nspose)

 leaky_re_lu_5 (LeakyReLU)   (None, 50, 50, 64)        0

 conv2d_transpose_1 (Conv2DT  (None, 100, 100, 64)     16448
 ranspose)

 leaky_re_lu_6 (LeakyReLU)   (None, 100, 100, 64)      0

 conv2d_transpose_2 (Conv2DT  (None, 300, 300, 64)     16448
 ranspose)

 leaky_re_lu_7 (LeakyReLU)   (None, 300, 300, 64)      0

 conv2d_4 (Conv2D)           (None, 300, 300, 3)       1731

=================================================================
Total params: 192,740,227
Trainable params: 192,740,227
Non-trainable params: 0
_________________________________________________________________
```

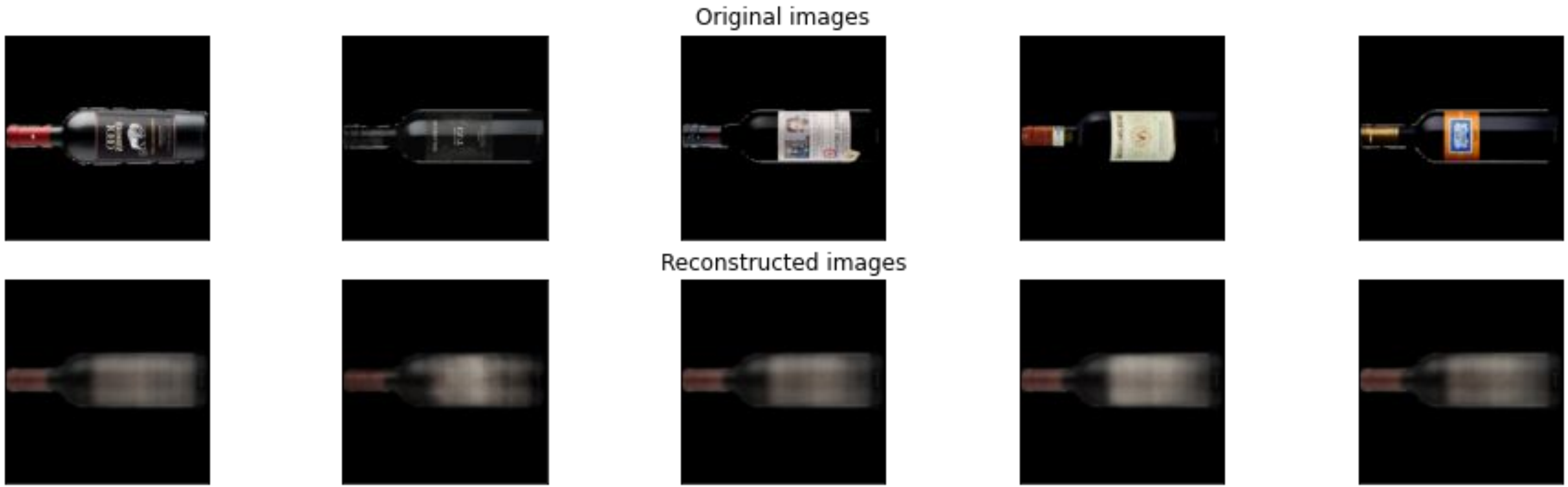
Discriminator

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 300, 300, 64)      832

 leaky_re_lu (LeakyReLU)     (None, 300, 300, 64)      0

 conv2d_1 (Conv2D)           (None, 100, 100, 128)     32896

 leaky_re_lu_1 (LeakyReLU)   (None, 100, 100, 128)     0

 conv2d_2 (Conv2D)           (None, 50, 50, 128)       65664

 leaky_re_lu_2 (LeakyReLU)   (None, 50, 50, 128)       0

 conv2d_3 (Conv2D)           (None, 50, 50, 256)       131328

 leaky_re_lu_3 (LeakyReLU)   (None, 50, 50, 256)       0

 flatten (Flatten)           (None, 640000)            0

 dropout (Dropout)           (None, 640000)            0

 dense (Dense)               (None, 1)                 640001

=================================================================
Total params: 870,721
Trainable params: 870,721
Non-trainable params: 0
_________________________________________________________________
```


VAE Loss over Epcohs
Train loss
Val loss
Log loss
Epochs
10^8
10^7
10^6
10^5
10^4
102 104 106 108 110 112 114 116 118 120 122 124 126 128 130 132 134 136 138 140 142 144 146 148 150 152 154 156 158 160 162 164 166 168 170 172 174 176 178 180 182 184 186 188 190 192 194 196 198 200


VAE Loss over Epcohs
Train loss
Val loss
Log loss
10^6
10^5
10^4
202
204
206
208
210
212
214
216
218
220
222
224
226
228
230
232
234
236
238
240
242
244
246
248
250
252
254
256
258
260
262
264
266
268
270
272
274
276
278
280
282
284
286
288
290
292
294
296
298
300
Epochs

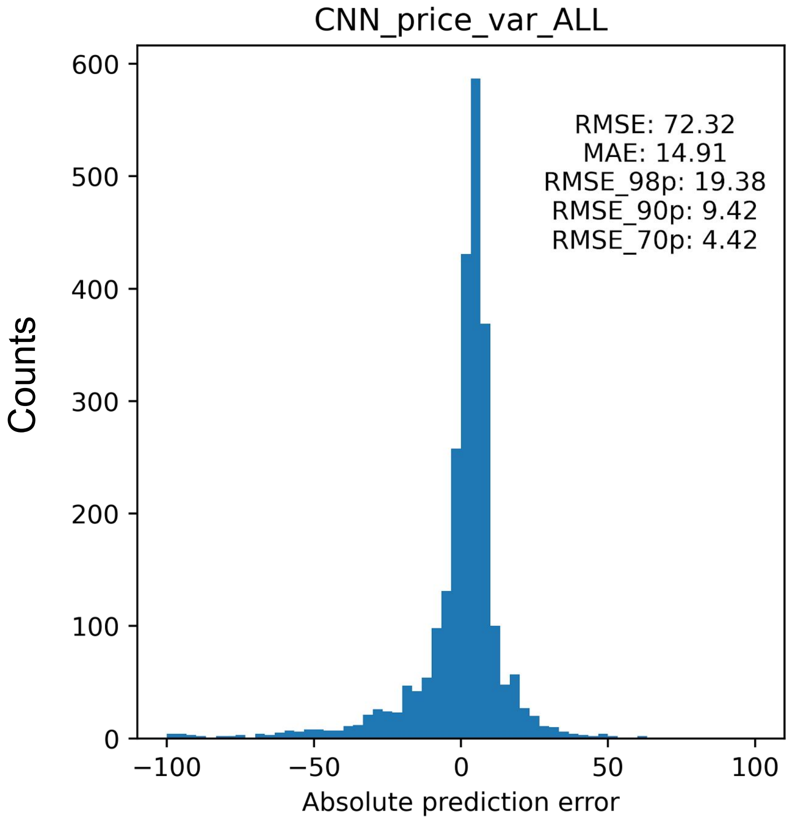
Epoch 139

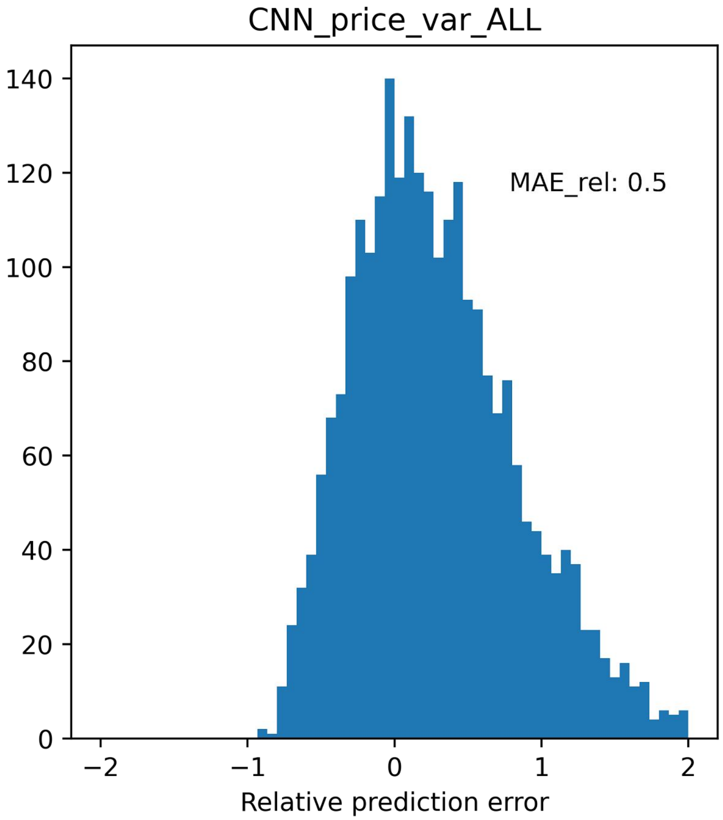


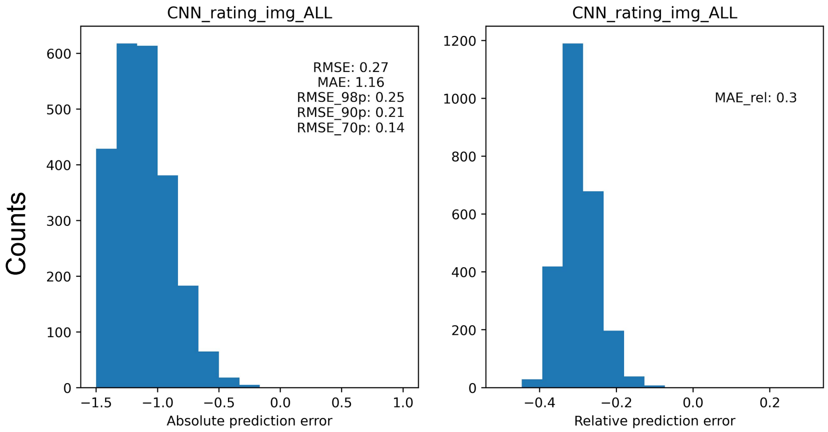
CNN Structure - Code

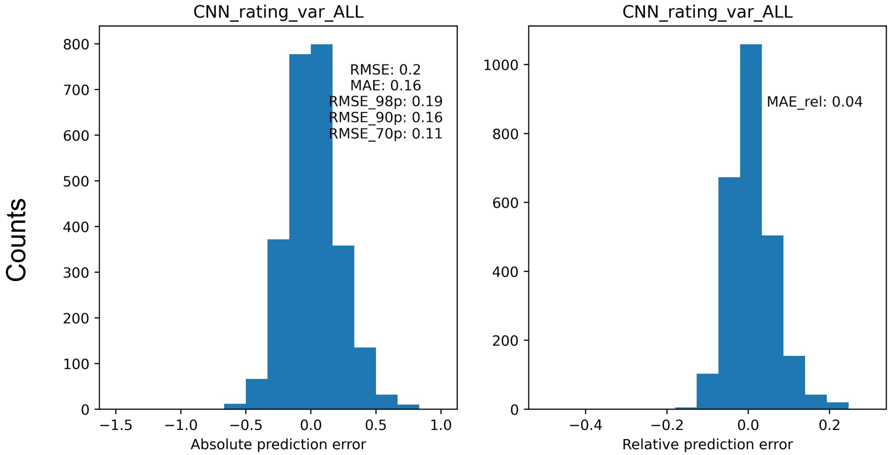
```
class Net(nn.Module):
    def __init__(self):
        super(Net,self).__init__()
        self.img_features_ =nn.Sequential(

            nn.Conv2d(4, 128, 5),
            nn.ReLU(),
            nn.Dropout(),
            nn.MaxPool2d(5,ceil_mode=True),
            nn.Conv2d(128, 64, 5),
            nn.ReLU(),
            nn.MaxPool2d(5,ceil_mode=True),
            nn.Flatten(),   #shape 432
            #nn.Linear(432, 1),
        )
        self.num_features_ = nn.Sequential(
            nn.Linear(6,128),
            nn.ReLU(),
            nn.Dropout(),
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Dropout(),      #shape 432
        )

        self.com_features_ = nn.Sequential(
            nn.Linear(9280,256),
            nn.ReLU(),
            nn.Dropout(),
            nn.Linear(256, 512),
            nn.ReLU(),
            nn.Linear(512, 256),
            nn.Linear(256,1),
        )


    def forward(self, x,y):
        x = self.img_features_(x)
        y = self.num_features_(y)
        z = torch.cat((x,y),1)
        z = self.com_features_(z)
        return z
```

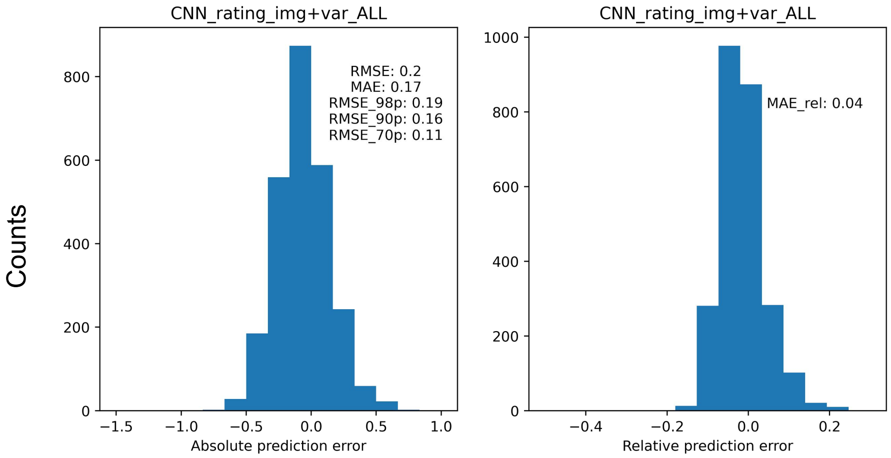
(C)NN regression residuals (predicting by variables)





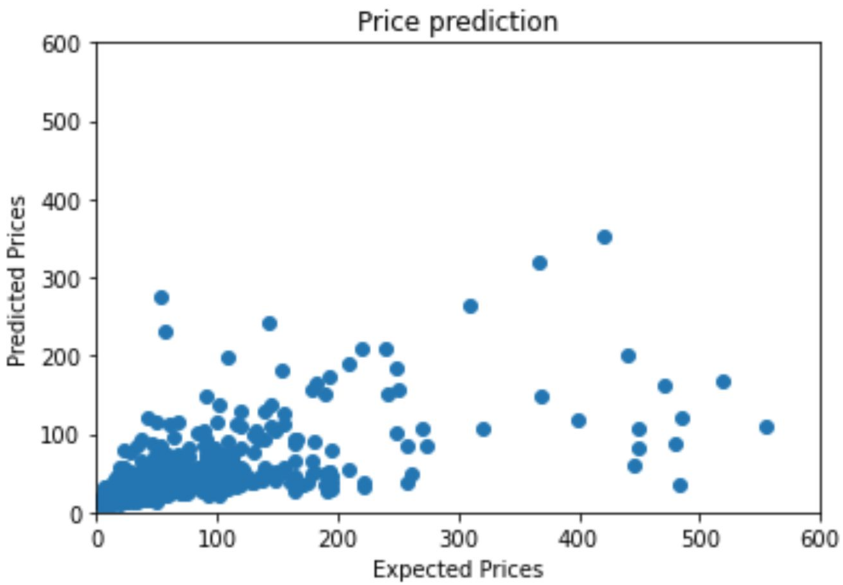

CNN_rating_img_ALL
RMSE: 0.27
MAE: 1.16
RMSE_98p: 0.25
RMSE_90p: 0.21
RMSE_70p: 0.14
Counts
Absolute prediction error
CNN_rating_img_ALL
MAE_rel: 0.3
Relative prediction error


CNN_rating_var_ALL
RMSE: 0.2
MAE: 0.16
RMSE_98p: 0.19
RMSE_90p: 0.16
RMSE_70p: 0.11
Counts
Absolute prediction error
CNN_rating_var_ALL
MAE_rel: 0.04
Relative prediction error

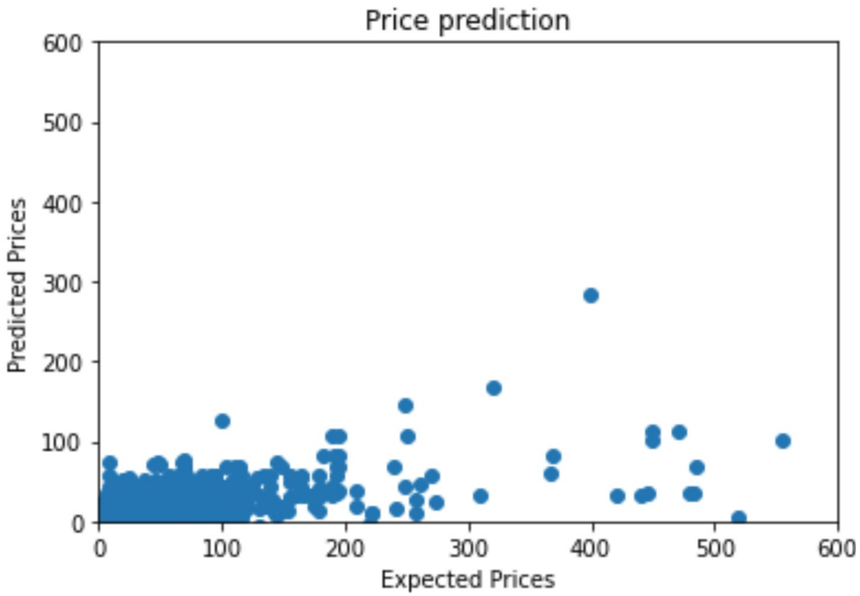

CNN_rating_img+var_ALL
RMSE: 0.2
MAE: 0.17
RMSE_98p: 0.19
RMSE_90p: 0.16
RMSE_70p: 0.11
Counts
Absolute prediction error
CNN_rating_img+var_ALL
MAE_rel: 0.04
Relative prediction error

CNN Evalutation Price Plots

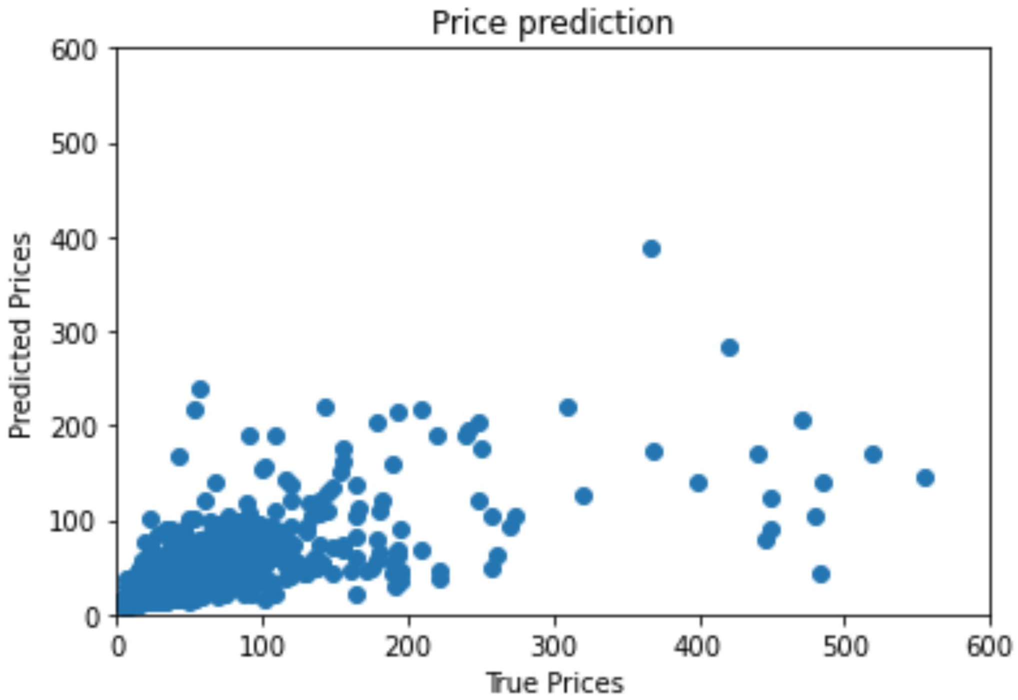
Variables and Images

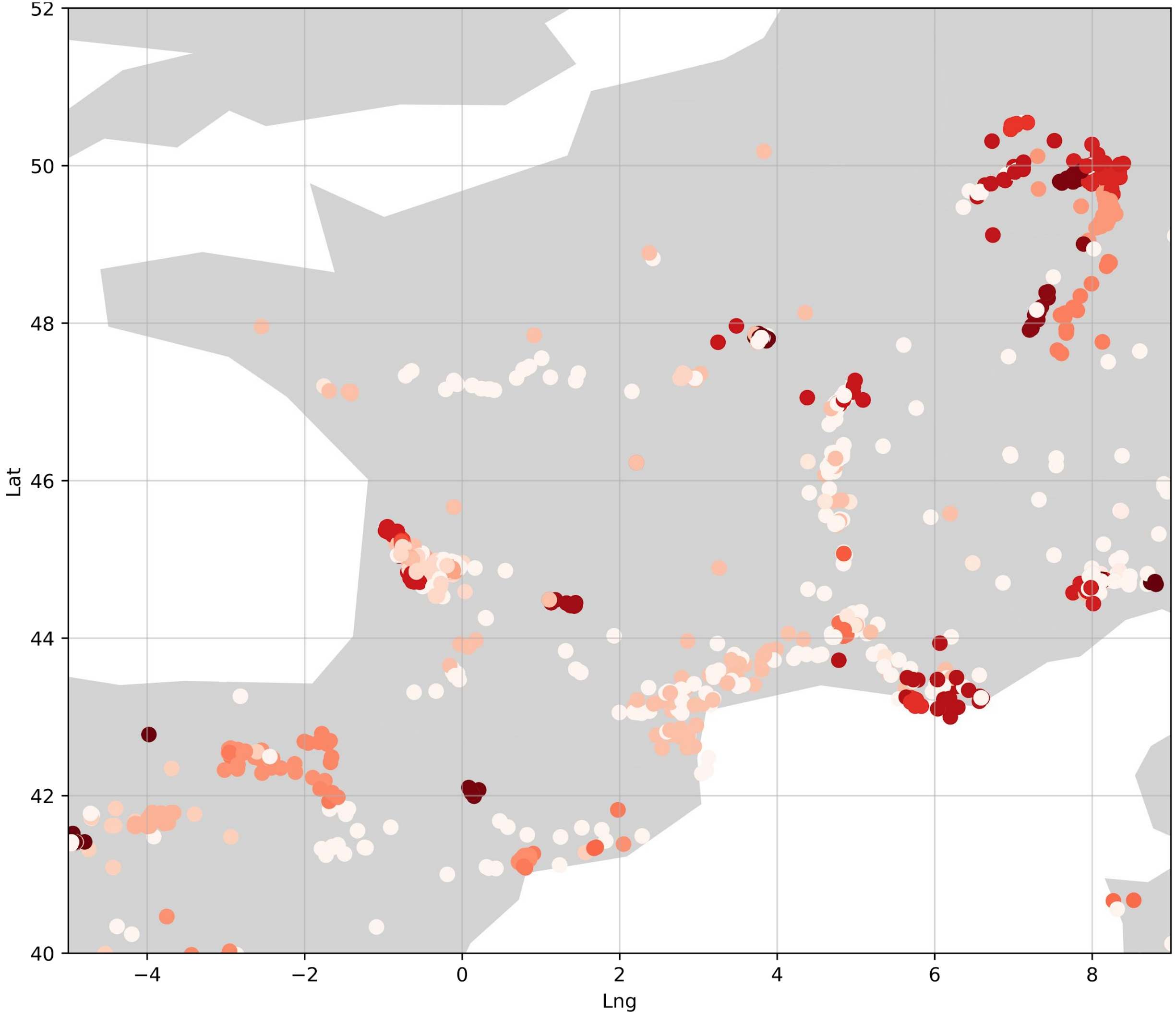


Images only

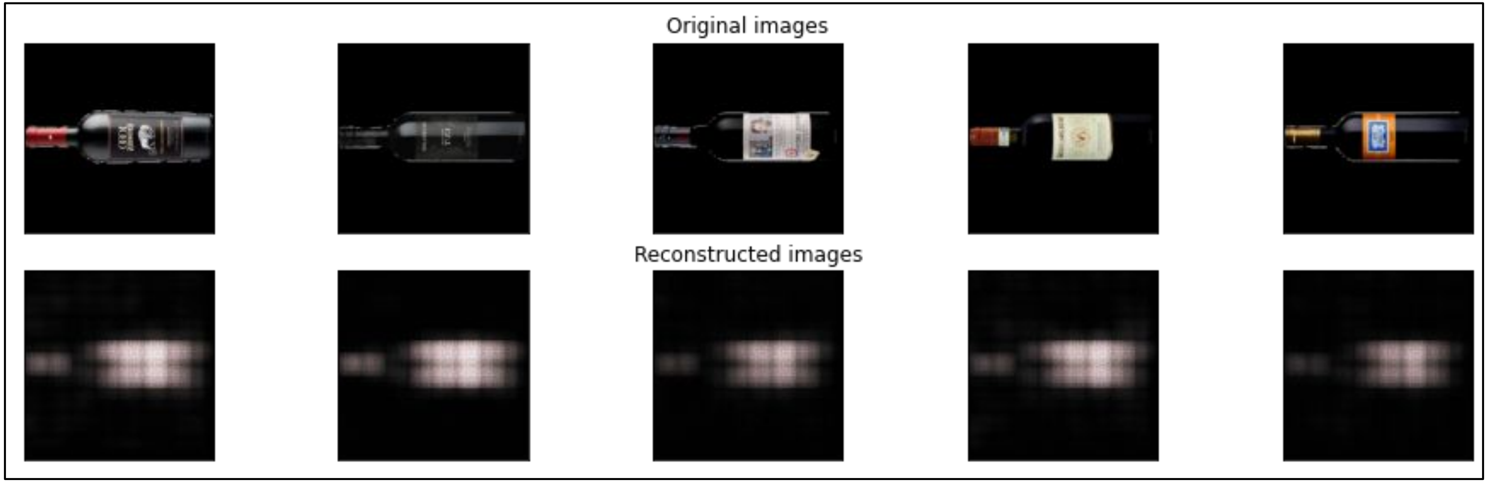


CNN Evaluation Price Plots - Variables Only




52
50
48
46
44
42
40
Lat
−4
−2
0
2
4
6
8
Lng

1 Epoch



21 Epoch