Python for Data Analysis
Final Project

YearPredictionMSD Data Set

CESAR Thomas
CHARRUEY Adrien
ESILV, A4, DIA 3

Summary

1. Introduction
2. Steps
3. Variables created
4. Conclusion

Introduction - Data Set

UCI Machine Learning Repository, YearPredictionMSD Data Set

Link : https://archive.ics.uci.edu/ml/datasets/YearPredictionMSD

Remark : This data is a subset of the Million Song Dataset:

http://labrosa.ee.columbia.edu/millionsong/

a collaboration between LabROSA (Columbia University) and The Echo Nest

Introduction - Data Set

This data set is composed of 515 345 songs released between 1922 and 2011.

These songs are mostly western, commercial tracks.

There are 91 variables.

The first variable is the release year, ranging from 1922 to 2011.

The 12 following variables are timbre averages, and the 78 last variables are timbre covariances. They take real values.

The dataset will be further explored in the "exploratory analysis and data-visualization" section

Introduction - Problem

We wish to predict the release year of a given song, from 90 of its audio features (the 12 timbre averages and the 78 timbre covariances).

Therefore, the target variable is the release year, and the features are the 12 timbre averages and the 78 timbre covariances.

This problem is a regression problem, as the target variable takes numerical values.

Steps

1. Data preprocessing
2. Exploratory data analysis and data visualization
3. Splitting, scaling and downsampling the data set
4. Feature selection
5. Modeling
6. Comparison between models, and model choice
7. API

Variables created

df : dataframe contenant tout le deatset

df_train : training data set

df_test : testing data set

df_train_s : scaled training data set

df_test_s : scaled testing data set

df_train_sampled : scaled and sampled training data set

df_test_10ft : testing dataframes scaled and with the target and the 10 most important features

df_test_20ft : testing dataframes scaled and with the target and the 20 most important features

df_train_10ft : training dataframe scaled, but not downsampled, with the target and the 10 most important features

df_train_20ft : training dataframe scaled, but not downsampled, with the target and the 20 most important features

df_train_samp_10ft : training dataframe scaled and downsampled, with the target and the 10 most important features

df_train_samp_20ft : training dataframe scaled and downsampled, with the target and the 20 most important features

df_metrics : the metrics of every model tested

model_names : list of all the models tried

datasets : list of all the datasets tried

list_preds : liste des predictions sur le testing data set pour chaque modèle

Conclusion

We tried the combination of 6 different models, and 4 different data sets (10 features and not downsampled, 20 features and not downsampled, 10 features and downsampled, 20 features and downsampled).

It took about 50 minutes to execute, and we got the following results :

Metrics results

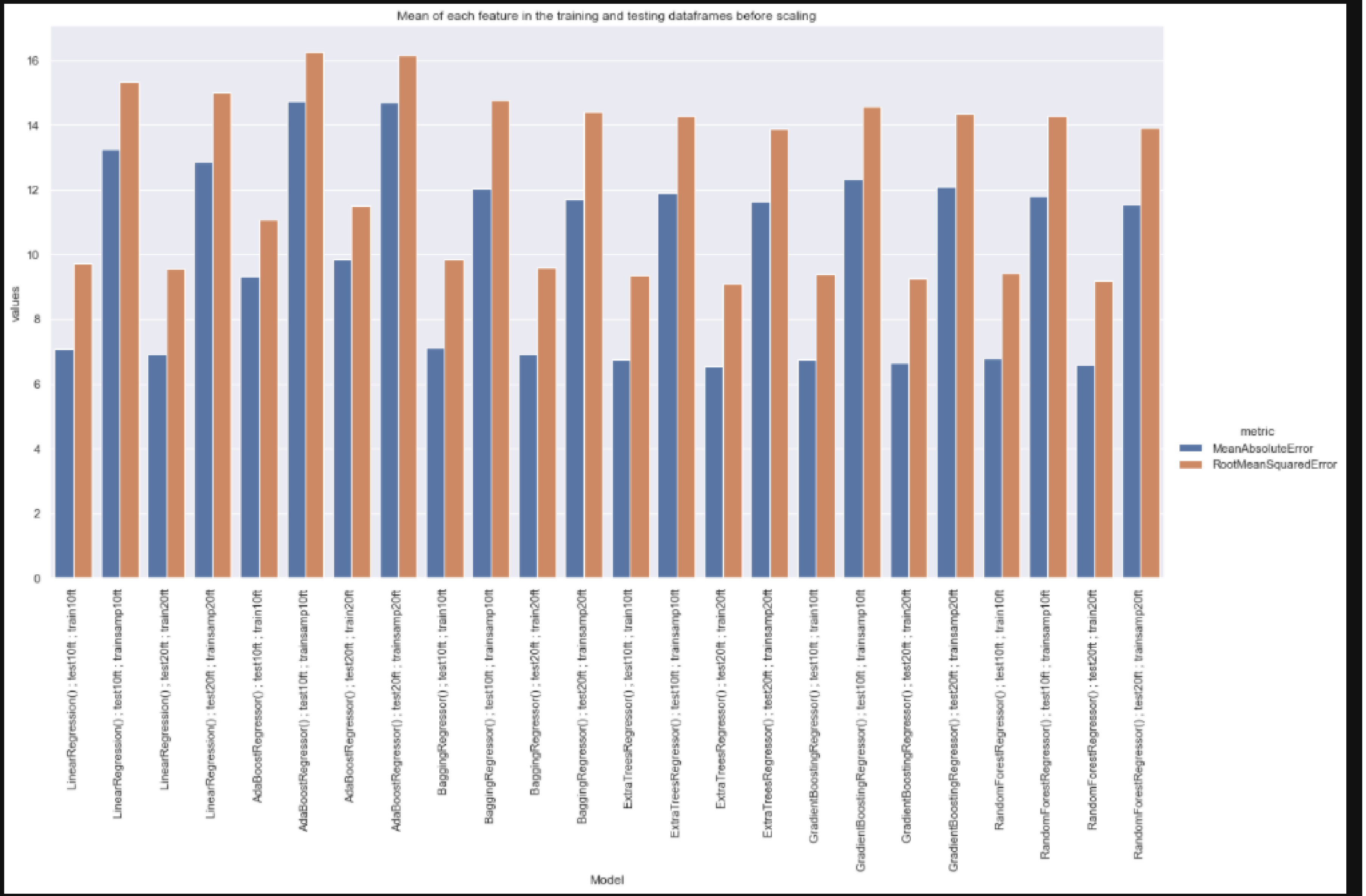
[50]:
```
df_metrics
```

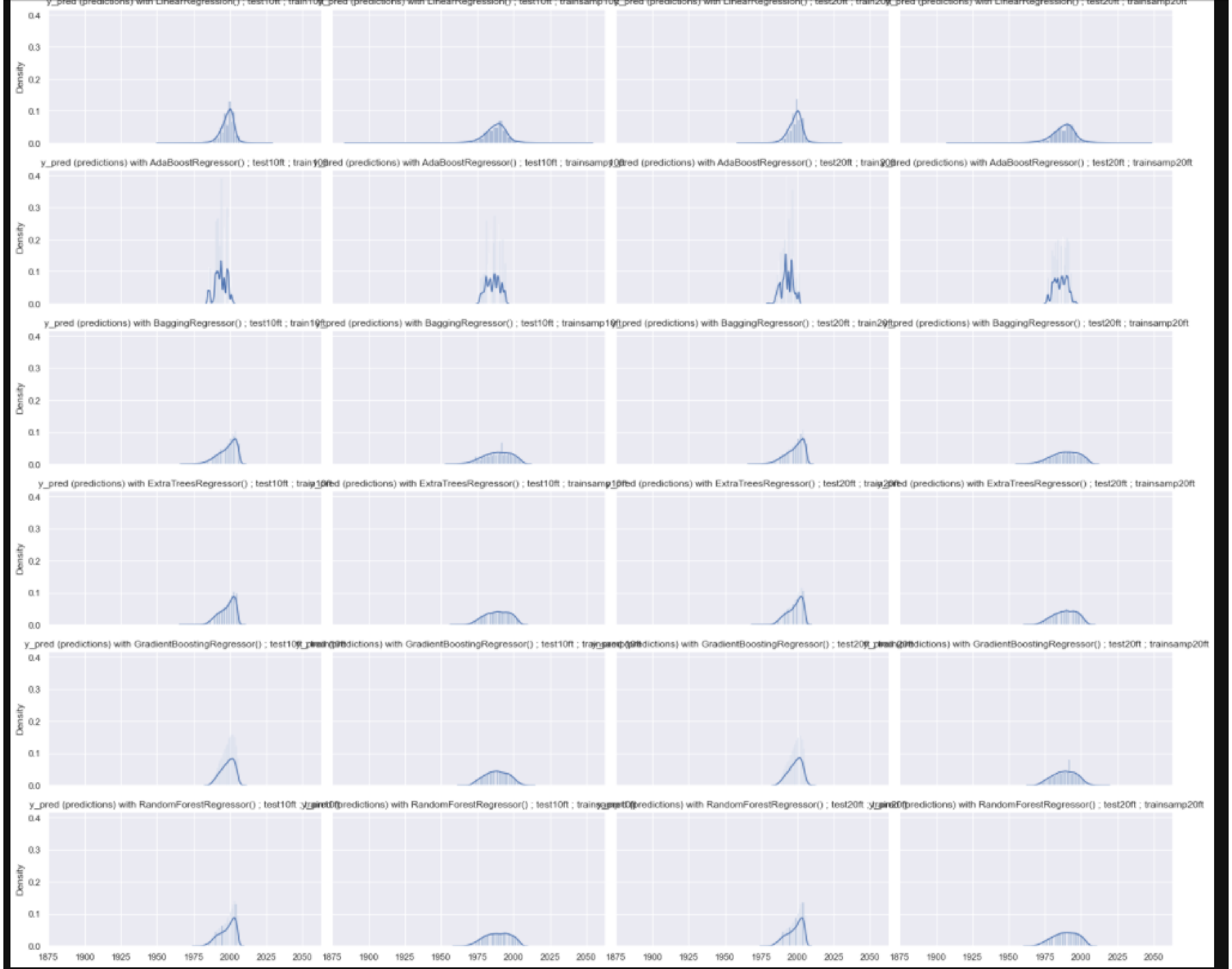
[50]:

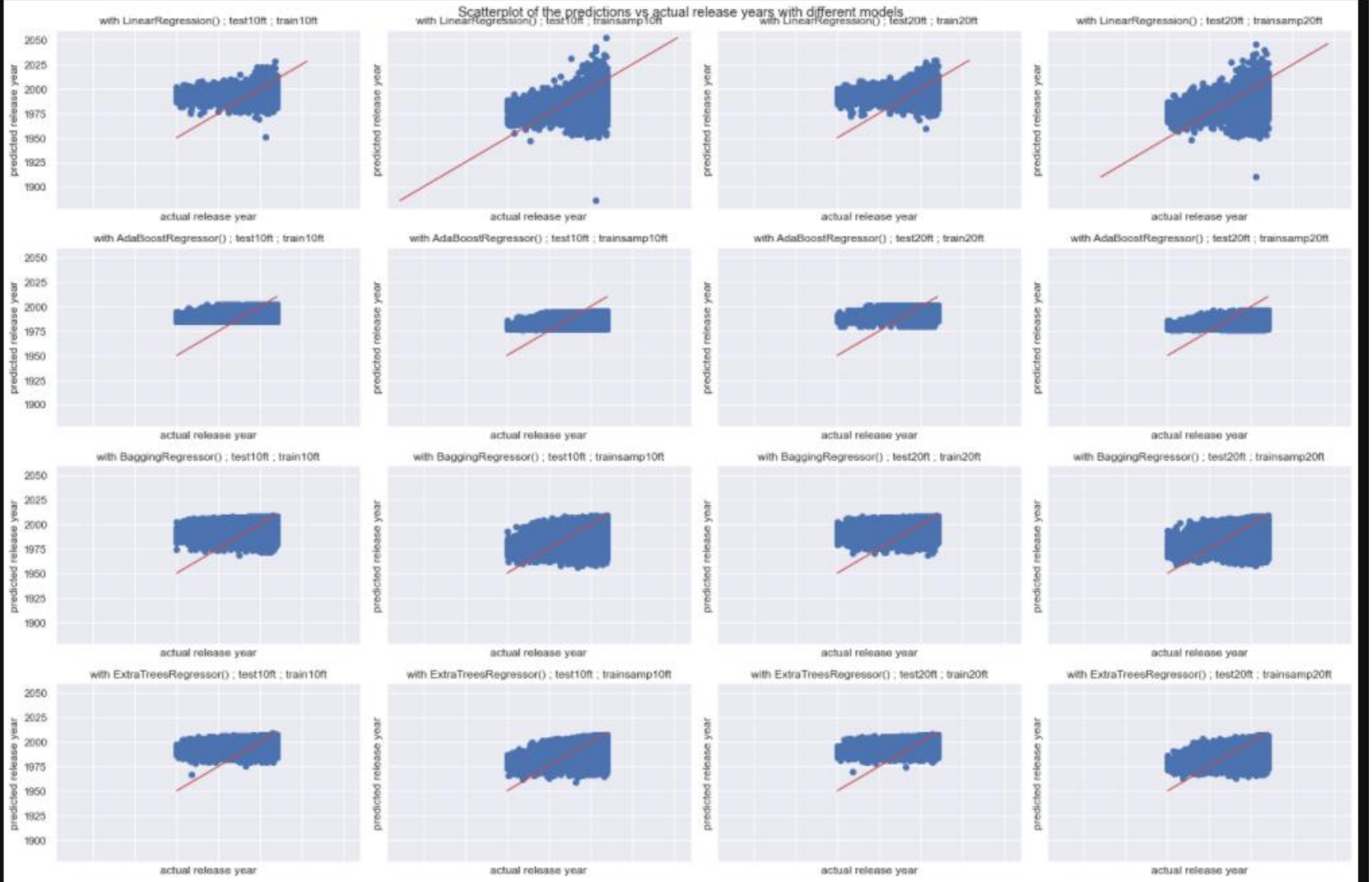
	Model	ExplainedVariance	MeanAbsoluteError	MeanSquaredError	RootMeanSquaredError	R^2	R^2adjusted
0	LinearRegression() ; test10ft ; train10ft	0.153845	7.066032	94.155367	9.703369	0.153787	0.152306
1	LinearRegression() ; test10ft ; trainsamp10ft	-0.028878	13.232304	235.073851	15.332118	-1.112706	-1.116402
2	LinearRegression() ; test20ft ; train20ft	0.180753	6.915865	91.161091	9.547832	0.180697	0.179264
3	LinearRegression() ; test20ft ; trainsamp20ft	-0.009589	12.853617	224.654339	14.988474	-1.019062	-1.022594
4	AdaBoostRegressor() ; test10ft ; train10ft	0.109437	9.313218	122.358080	11.061559	-0.099683	-0.101607
5	AdaBoostRegressor() ; test10ft ; trainsamp10ft	0.098540	14.720312	263.573269	16.234940	-1.368842	-1.372986
6	AdaBoostRegressor() ; test20ft ; train20ft	0.111932	9.853753	132.181058	11.497002	-0.187966	-0.190044
7	AdaBoostRegressor() ; test20ft ; trainsamp20ft	0.106684	14.688082	261.095603	16.158453	-1.346574	-1.350680
8	BaggingRegressor() ; test10ft ; train10ft	0.132678	7.097117	96.758363	9.836583	0.130392	0.128871
9	BaggingRegressor() ; test10ft ; trainsamp10ft	-0.133459	12.024100	218.010556	14.765181	-0.959351	-0.962779
10	BaggingRegressor() ; test20ft ; train20ft	0.174409	6.901467	92.105751	9.597174	0.172207	0.170759
11	BaggingRegressor() ; test20ft ; trainsamp20ft	-0.077511	11.709989	207.111417	14.391366	-0.861396	-0.864653
12	ExtraTreesRegressor() ; test10ft ; train10ft	0.216441	6.756772	87.467149	9.352387	0.213896	0.212521
13	ExtraTreesRegressor() ; test10ft ; trainsamp10ft	0.022049	11.886429	203.709038	14.272668	-0.830818	-0.834021
14	ExtraTreesRegressor() ; test20ft ; train20ft	0.257864	6.562384	82.853986	9.102416	0.255357	0.254054
15	ExtraTreesRegressor() ; test20ft ; trainsamp20ft	0.090663	11.636739	192.860971	13.887439	-0.733322	-0.736354
16	GradientBoostingRegressor() ; test10ft ; train...	0.208792	6.751067	88.044610	9.383209	0.208707	0.207322
17	GradientBoostingRegressor() ; test10ft ; train...	0.020603	12.324802	212.348669	14.572188	-0.908466	-0.911804
18	GradientBoostingRegressor() ; test20ft ; train...	0.231764	6.642910	85.489891	9.246074	0.231667	0.230323
19	GradientBoostingRegressor() ; test20ft ; train...	0.044055	12.090539	205.512069	14.335692	-0.847022	-0.850253
20	RandomForestRegressor() ; test10ft ; train10ft	0.207035	6.769229	88.475415	9.406137	0.204835	0.203444
21	RandomForestRegressor() ; test10ft ; trainsamp...	-0.003995	11.785839	203.627503	14.269811	-0.830085	-0.833286
22	RandomForestRegressor() ; test20ft ; train20ft	0.245743	6.583476	84.165438	9.174172	0.243570	0.242247
23	RandomForestRegressor() ; test20ft ; trainsamp...	0.060365	11.524798	193.267037	13.902052	-0.736971	-0.740010

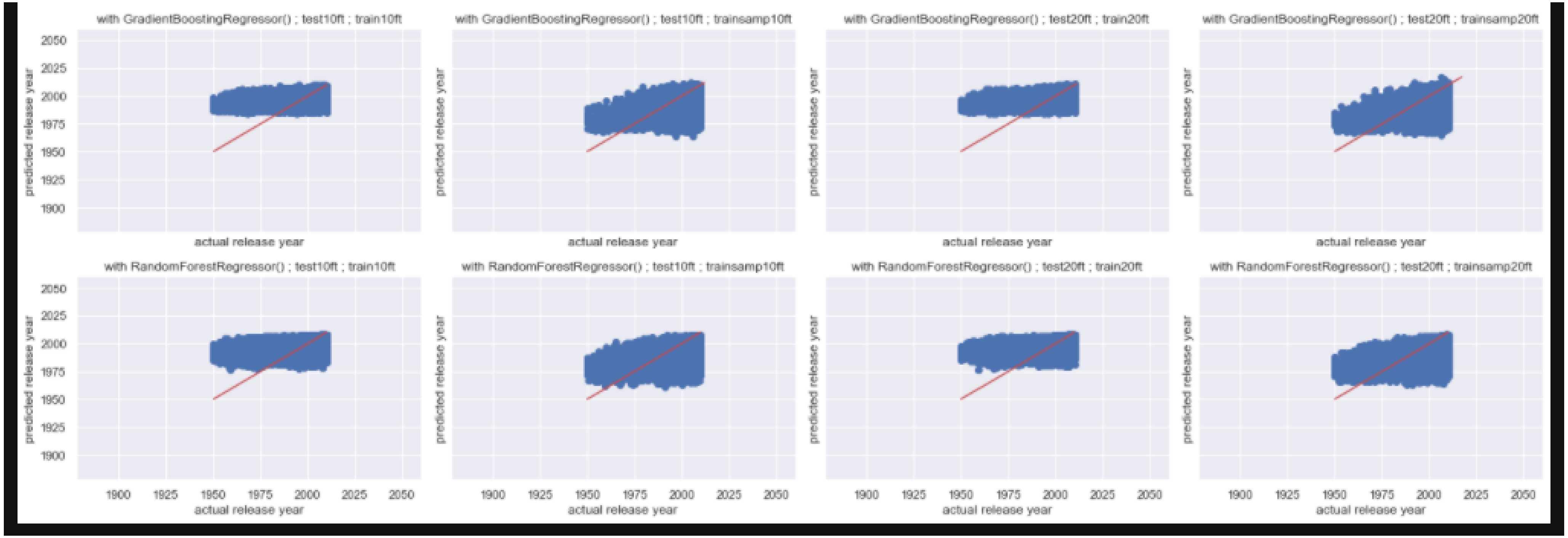
Comparing MAE and RMSE of each model :

```
[51]: compare = pd.melt(df_metrics[['Model', 'MeanAbsoluteError', 'RootMeanSquaredError']], id_vars="Model", var_name="metric", value_name="values
      sns.catplot(x='Model', y='values', hue='metric', data=compare, kind='bar', height=8, aspect=2)
      plt.xticks(rotation = 90)
      plt.title('Mean of each feature in the training and testing dataframes before scaling');
```




y_pred (predictions) with AdaBoostRegressor() ; test10ft ; train10ft
y_pred (predictions) with AdaBoostRegressor() ; test10ft ; trainsamp10ft
y_pred (predictions) with AdaBoostRegressor() ; test20ft ; train20ft
y_pred (predictions) with AdaBoostRegressor() ; test20ft ; trainsamp20ft
y_pred (predictions) with BaggingRegressor() ; test10ft ; train10ft
y_pred (predictions) with BaggingRegressor() ; test10ft ; trainsamp10ft
y_pred (predictions) with BaggingRegressor() ; test20ft ; train20ft
y_pred (predictions) with BaggingRegressor() ; test20ft ; trainsamp20ft
y_pred (predictions) with ExtraTreesRegressor() ; test10ft ; train10ft
y_pred (predictions) with ExtraTreesRegressor() ; test10ft ; trainsamp10ft
y_pred (predictions) with ExtraTreesRegressor() ; test20ft ; train20ft
y_pred (predictions) with ExtraTreesRegressor() ; test20ft ; trainsamp20ft
y_pred (predictions) with GradientBoostingRegressor() ; test10ft ; train10ft
y_pred (predictions) with GradientBoostingRegressor() ; test10ft ; trainsamp10ft
y_pred (predictions) with GradientBoostingRegressor() ; test20ft ; train20ft
y_pred (predictions) with GradientBoostingRegressor() ; test20ft ; trainsamp20ft
y_pred (predictions) with RandomForestRegressor() ; test10ft ; train10ft
y_pred (predictions) with RandomForestRegressor() ; test10ft ; trainsamp10ft
y_pred (predictions) with RandomForestRegressor() ; test20ft ; train20ft
y_pred (predictions) with RandomForestRegressor() ; test20ft ; trainsamp20ft
Density
0.0
0.1
0.2
0.3
0.4
1875
1900
1925
1950
1975
2000
2025
2050


Scatterplot of the predictions vs actual release years with different models
with LinearRegression() ; test10ft ; train10ft
with LinearRegression() ; test10ft ; trainsamp10ft
with LinearRegression() ; test20ft ; train20ft
with LinearRegression() ; test20ft ; trainsamp20ft
with AdaBoostRegressor() ; test10ft ; train10ft
with AdaBoostRegressor() ; test10ft ; trainsamp10ft
with AdaBoostRegressor() ; test20ft ; train20ft
with AdaBoostRegressor() ; test20ft ; trainsamp20ft
with BaggingRegressor() ; test10ft ; train10ft
with BaggingRegressor() ; test10ft ; trainsamp10ft
with BaggingRegressor() ; test20ft ; train20ft
with BaggingRegressor() ; test20ft ; trainsamp20ft
with ExtraTreesRegressor() ; test10ft ; train10ft
with ExtraTreesRegressor() ; test10ft ; trainsamp10ft
with ExtraTreesRegressor() ; test20ft ; train20ft
with ExtraTreesRegressor() ; test20ft ; trainsamp20ft
predicted release year
actual release year
2050
2025
2000
1975
1950
1925
1900


with GradientBoostingRegressor() ; test10ft ; train10ft
with GradientBoostingRegressor() ; test10ft ; trainsamp10ft
with GradientBoostingRegressor() ; test20ft ; train20ft
with GradientBoostingRegressor() ; test20ft ; trainsamp20ft
with RandomForestRegressor() ; test10ft ; train10ft
with RandomForestRegressor() ; test10ft ; trainsamp10ft
with RandomForestRegressor() ; test20ft ; train20ft
with RandomForestRegressor() ; test20ft ; trainsamp20ft
predicted release year
actual release year
1900
1925
1950
1975
2000
2025
2050

We can see that the model with the best Root Mean Squared Error is ExtraTreesRegressor, with the sets of 20 features, and the training set not downsampled. But to have a model small enough to load in a reasonnable time, and not to many features for the user to enter in the API, we decided to keep the best model with data sets of 10 features. It was the ExtraTreesRegressor, with the sets of 10 features, and the training set not downsampled. Unfortunately we got this memory error :

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_9312/1908189703.py in <module>
----> 1 result = ValuePredictor([0.03, 0.03, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0])

~\AppData\Local\Temp/ipykernel_9312/1947034984.py in ValuePredictor(to_predict_list)
      1 def ValuePredictor(to_predict_list):
      2     to_predict = np.array(to_predict_list).reshape(1,10)
----> 3     loaded_model = pickle.load(open("model.pkl", "rb"))
      4     result = loaded_model.predict(to_predict)
      5     return result[0]

sklearn\tree\_tree.pyx in sklearn.tree._tree.Tree.__setstate__()

sklearn\tree\_tree.pyx in sklearn.tree._tree.Tree._resize_c()

sklearn\tree\_utils.pyx in sklearn.tree._utils.safe_realloc()

MemoryError: could not allocate 45935176 bytes
```

So we kept the second best model with data sets of 10 features, which was the GradientBoostingRegressor, with the sets of 10 features, and the training set not downsampled. It has a RMSE of 9,38.
The API done with flask gives the following :

Year Prediction MSD Data Set

"Please enter values for each feature"

feature 1	-0,000001
feature 2	-0,000038
feature 3	0,000025
feature 4	0,00002
feature 5	0,000015
feature 6	-0,000011
feature 7	0,000011
feature 10	0,000014
feature 12	-0,000018
feature 20	0,000049

Submit

Result : **1996**