SongEnduranceNotebook

December 7, 2018

DISCLAIMER : There are lot of experiments done in between and hence the code will not run startight end to end

In [ ]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-py
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list the fi

import os
print(os.listdir("../input"))

# Any results you write to the current directory are saved as output.
```

In [ ]:
```
df = pd.read_csv('../input/clean-songs/song_data.csv')
```

In [ ]:
```
import seaborn as sns
train_corr = df.copy()
train_corr = train_corr.drop('log_youtube',axis=1)
train_corr = train_corr.drop('log_spotify',axis=1)
train_corr = train_corr.drop('TopSongsArtist',axis=1)
train_corr = train_corr.drop('TopSongsArtist10',axis=1)
train_corr = train_corr.drop('TopSongsArtist100',axis=1)
train_corr = train_corr.drop('time_signature',axis=1)
train_corr = train_corr.drop('mode',axis=1)
train_corr = train_corr.drop('key',axis=1)
train_corr = train_corr.drop('song decay rate',axis=1)
train_corr = train_corr.drop('current popularity',axis=1)
train_corr = train_corr.drop('diff_pop',axis=1)
train_corr = train_corr.drop('Artist_lifetime_grammy_achievement',axis=1)
train_corr = train_corr.drop('Artist_grammy_nominations',axis=1)
# train_corr = train_corr.drop('diff_pop',axis=1)

plt.figure(figsize=(25,25))
```

```
        sns.heatmap(train_corr.corr(method="pearson"), annot=True, annot_kws={"size": 12},cmap=
        # train_corr.corr(method='pearson').style.format("{:.2}").background_gradient(cmap=plt
```

In [ ]:
```
df = df[df['Youtube viewcount']>1]
df = df[df['Popularity']>0]
df['log_youtube'] = np.log(1+df['Youtube viewcount'])
df['log_spotify'] = np.log(1+df['Popularity'])
df['net_current_popularity'] = df['log_youtube']*df['log_spotify']
df['original_popularity'] = 136-df['Peak_Position']
df['current popularity']=0.0
df['song decay rate'] = 0.0
```

In [ ]:
```
df['diff_pop'] = df['net_current_popularity'] - df['original_popularity']
```

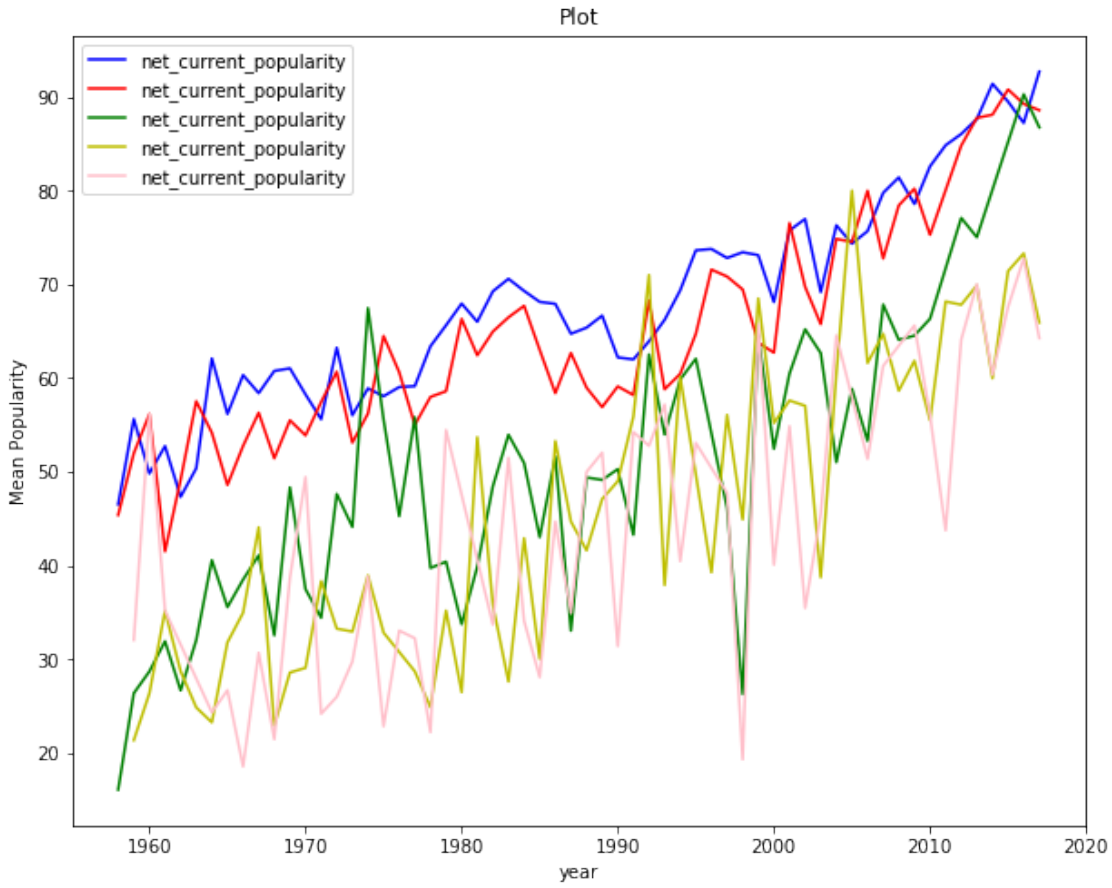
In [ ]:
```
df_base = df.copy()
```

In [ ]:
```
df_base['date'], df_base['month'],df_base['year'] = df_base['Entry_Date'].str.split('/
df_base['year'] = df_base['year'].apply(lambda x: ('19'+x) if int(x)>18 else ('20'+x))
df_base["year"] = pd.to_numeric(df_base["year"])
df_base['Artist_Name'] = df_base['Artist']
df_base.head()
```

In [ ]:
```
df_base = df_base.sort_values(by=['year'], ascending = True)
df_base.head()
```

In [59]:
```
df_peak1 = df_base[df_base['Peak_Position']==1]
df_peak2 = df_base[df_base['Peak_Position']==2]
df_peak25 = df_base[df_base['Peak_Position']==25]
df_peak50 = df_base[df_base['Peak_Position']==50]
df_peak75 = df_base[df_base['Peak_Position']==80]
```

In [60]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,8))
df_peak1.groupby('year')['net_current_popularity'].mean().sort_index().plot(color =
df_peak2.groupby('year')['net_current_popularity'].mean().sort_index().plot(color =
df_peak25.groupby('year')['net_current_popularity'].mean().sort_index().plot(color =
df_peak50.groupby('year')['net_current_popularity'].mean().sort_index().plot(color =
df_peak75.groupby('year')['net_current_popularity'].mean().sort_index().plot(color =
plt.legend()
plt.title('Plot');
plt.ylabel('Mean Popularity')
plt.xlabel('year')
plt.show()
```



LGBM ML Model

```
In [ ]: df_base.dtypes
```

```
In [ ]: df_1970=df_base.copy()
        df_1970x = df_base.copy()
        df_1970=df_1970[df_1970['year']==1970]
        df_1970=df_1970=df_1970.sort_values(by='net_current_popularity',ascending = False)
        df_1970.head(20)
```

```
In [71]: df_lgbm = df_base.copy()
```

```
In [72]: df_lgbm = df_lgbm.drop('Title',axis=1)
         df_lgbm = df_lgbm.drop('Popularity',axis=1)
         df_lgbm = df_lgbm.drop('original_popularity',axis=1)
         df_lgbm = df_lgbm.drop('log_youtube',axis=1)
         df_lgbm = df_lgbm.drop('log_spotify',axis=1)
         df_lgbm = df_lgbm.drop('Youtube viewcount',axis=1)
         df_lgbm = df_lgbm.drop('Entry_Date',axis=1)
         df_lgbm = df_lgbm.drop('Artist_Name',axis=1)
         df_lgbm = df_lgbm.drop('date',axis=1)
```

```
df_lgbm = df_lgbm.drop('month',axis=1)
df_lgbm = df_lgbm.drop('diff_pop',axis=1)
df_lgbm = df_lgbm.drop('Oscars_won',axis=1)
df_lgbm = df_lgbm.drop('current popularity',axis=1)
df_lgbm = df_lgbm.drop('mode',axis=1)
df_lgbm = df_lgbm.drop('song decay rate',axis=1)
df_lgbm = df_lgbm.drop('duration_ms',axis=1)
df_lgbm = df_lgbm.drop('Artist',axis=1)
# df_lgbm = df_lgbm.drop('Artist_lifetime_grammy_achievement',axis=1)
df_lgbm = df_lgbm.drop('time_signature',axis=1)
# df_lgbm = df_lgbm.drop('Artist_grammy_nominations',axis=1)
# df_lgbm = df_lgbm.drop('mode',axis=1)
# df_lgbm = df_lgbm.drop('key',axis=1)
```

In [73]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
df_lgbm['Artist'] = le.fit_transform(df_lgbm.Artist.values)
```

```
---------------------------------------------------------------------------

AttributeError                            Traceback (most recent call last)

<ipython-input-73-eefb309206ce> in <module>()
  1 from sklearn import preprocessing
  2 le = preprocessing.LabelEncoder()
----> 3 df_lgbm['Artist'] = le.fit_transform(df_lgbm.Artist.values)


/opt/conda/lib/python3.6/site-packages/pandas/core/generic.py in __getattr__(self, name
4374             if self._info_axis._can_hold_identifiers_and_holds_name(name):
4375                 return self[name]
-> 4376             return object.__getattribute__(self, name)
4377
4378     def __setattr__(self, name, value):


AttributeError: 'DataFrame' object has no attribute 'Artist'
```

In [ ]:
```
### FOR 1970
from sklearn.model_selection import train_test_split
import lightgbm as lgb
features = [c for c in df_lgbm.columns ]
features.remove("net_current_popularity")
features.remove("Artist_Name")
features.remove("Title")
```

```
train_x = df_lgbm[df_lgbm['year']!=1970]
train_y = train_x['net_current_popularity']
train_x = train_x[features]

valid_x = df_lgbm[df_lgbm['year']==1970]
valid_y = valid_x['net_current_popularity']
df_dummy1970 = valid_x.copy()
valid_x = valid_x[features]

# train_x, valid_x, train_y, valid_y = train_test_split(df_lgbm[features], df_lgbm["ne
lgb_params = {"objective" : "regression", "metric" : "rmse",
              "num_leaves" : 100, "learning_rate" : 0.02,
              "bagging_fraction" : 0.75, "feature_fraction" : 0.8, "bagging_frequency"

lgb_train = lgb.Dataset(train_x, label=train_y)
lgb_val = lgb.Dataset(valid_x, label=valid_y)
model = lgb.train(lgb_params, lgb_train, 500, valid_sets=[lgb_val], early_stopping_rou
```

In [ ]:
```
df_dummy1970['net_current_popularity'] = model.predict(valid_x[features], num_iteratio
```

In [45]:
```
df_dummy1970=df_dummy1970.sort_values(by='net_current_popularity',ascending = False)
df_dummy1970.head(20)
```

Out[45]:

	Title	...
12620	Immigrant Song	...
2266	Paranoid	...
17528	The Long And Winding Road	...
9300	War	...
9323	Wigwam	...
12607	We've Only Just Begun	...
8737	Make It With You	...
11100	Cracklin' Rosie	...
12601	I Think I Love You	...
1047	ABC	...
1103	Cecilia	...
12615	Let's Work Together	...
4638	The Love You Save	...
12613	See Me Feel Me	...
1082	Little Green Bag	...
6008	Your Song	...
12602	One Less Bell To Answer	...
11540	Lola	...
12600	Tears Of A Clown	...
19595	Mama Told Me (Not To Come)	...

[20 rows x 25 columns]

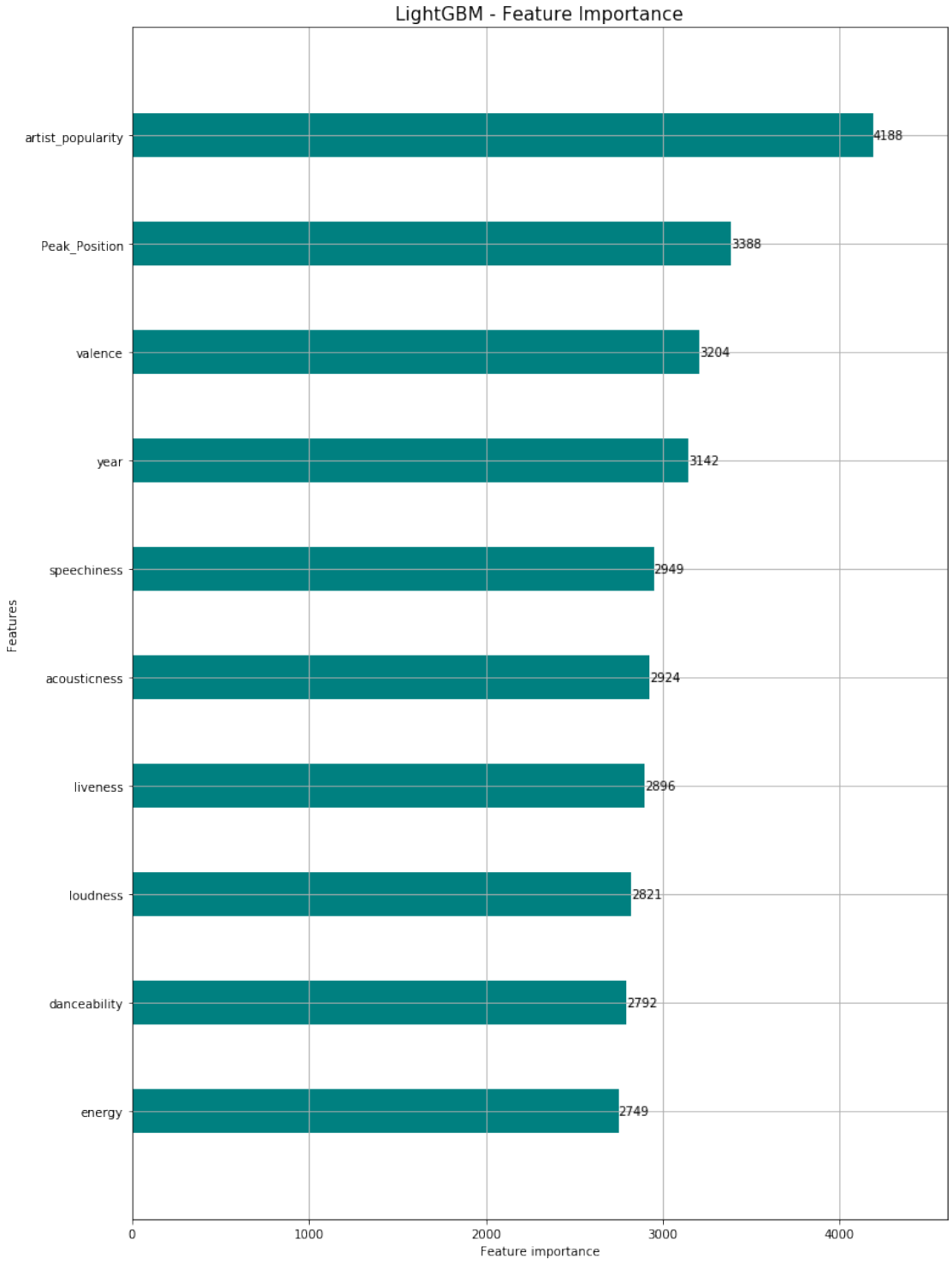
In [46]:
```
from sklearn.model_selection import train_test_split
import lightgbm as lgb
```

```
        features = [c for c in df_lgbm.columns ]
        features.remove("net_current_popularity")
        features.remove("Artist_Name")
        features.remove("Title")


        train_x, valid_x, train_y, valid_y = train_test_split(df_lgbm[features], df_lgbm["net_


        lgb_params = {"objective" : "regression", "metric" : "rmse",
                      "num_leaves" : 100, "learning_rate" : 0.02,
                      "bagging_fraction" : 0.75, "feature_fraction" : 0.8, "bagging_frequency"

        lgb_train = lgb.Dataset(train_x, label=train_y)
        lgb_val = lgb.Dataset(valid_x, label=valid_y)
        model = lgb.train(lgb_params, lgb_train, 500, valid_sets=[lgb_val], early_stopping_rou
```

```
Training until validation scores don't improve for 300 rounds.
[100]        valid_0's rmse: 10.3042
[200]        valid_0's rmse: 9.69139
[300]        valid_0's rmse: 9.60022
[400]        valid_0's rmse: 9.57557
[500]        valid_0's rmse: 9.56715
Did not meet early stopping. Best iteration is:
[496]        valid_0's rmse: 9.56641
```

```
In [ ]: from xgboost import XGBRegressor
        from sklearn import metrics
        train_x, valid_x, train_y, valid_y = train_test_split(df_lgbm[features], df_lgbm["net_
        my_model = XGBRegressor(n_estimators=5000, learning_rate=0.02)
        my_model.fit(train_x, train_y, early_stopping_rounds=300,
                     eval_set=[(valid_x, valid_y)], verbose=False)
        preds = my_model.predict(valid_x)
        print('\nroot Mean Square error" ', np.sqrt(metrics.mean_squared_error(valid_y,preds)))
```

```
In [50]: fig, ax = plt.subplots(figsize=(12,18))
         lgb.plot_importance(model, max_num_features=10, height=0.4, ax=ax,color='teal')
         ax.grid(False)
         plt.title("LightGBM - Feature Importance", fontsize=15)
         #plt.ylabel(fontsize=10)
         plt.grid()
         plt.show()
```




```
In [ ]: import xgboost as xgb
        fig, ax = plt.subplots(figsize=(12,18))
        xgb.plot_importance(my_model, max_num_features=10, height=0.4, ax=ax)
```

```
        plt.show()
```

```
In [ ]: df_lgbm.head()
```

```
In [78]: from sklearn import metrics
         from sklearn.model_selection import train_test_split
         import lightgbm as lgb
         peak_list = df_lgbm['Peak_Position'].unique().tolist()
         peak_list.sort()
         features = [c for c in df_lgbm.columns ]
         features.remove("net_current_popularity")
         # print (range(len(year_list)))
         # for i in range(0,100):
         #      del peak_list[i]
         rmse_list=[]
         for y in peak_list:
             print (y)
             df_year_ts = df_lgbm[df_lgbm['Peak_Position']==y]
             df_year_tr = df_lgbm[df_lgbm['year']!=y]
         #     train_x, valid_x, train_y, valid_y = train_test_split(df_year_tr[features],df_y
             lgb_params = {"objective" : "regression", "metric" : "rmse",
                           "num_leaves" : 100, "learning_rate" : 0.02,
                           "bagging_fraction" : 0.75, "feature_fraction" : 0.8, "bagging_freque

             lgb_train = lgb.Dataset(df_year_tr[features], label=df_year_tr['net_current_popula
             lgb_val = lgb.Dataset(df_year_ts[features], label=df_year_ts['net_current_popular
             model = lgb.train(lgb_params, lgb_train, 500, valid_sets=[lgb_val], early_stopping

             pred_test_y = model.predict(df_year_ts[features], num_iteration=model.best_iterati
             lrmse = np.sqrt(metrics.mean_squared_error(pred_test_y, df_year_ts['net_current_po
             rmse_list.append(lrmse)

         #     my_model = XGBRegressor(n_estimators=5000, learning_rate=0.02)
         #     my_model.fit(df_year_tr[features], df_year_tr['net_current_popularity'], early_
         #              eval_set=[(df_year_ts[features], df_year_ts['net_current_popularity'])]
         #     preds = my_model.predict(df_year_ts[features])
         #     rmse_list.append(np.sqrt(metrics.mean_squared_error(df_year_ts['net_current_pop
```

```
1
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 5.83105
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 5.50562
2
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 5.97805
Did not meet early stopping. Best iteration is:
```

```
[500]        valid_0's rmse: 5.68558
3
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.61735
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.28442
4
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.59035
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.1977
5
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 5.69217
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 5.38686
6
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.35446
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 5.99852
7
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.90183
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.58173
8
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.01819
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.66169
9
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.96315
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.618
10
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.00983
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 5.68136
11
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 5.96303
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 5.66459
12
Training until validation scores don't improve for 200 rounds.
```

```
[400]        valid_0's rmse: 6.74953
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.46479
13
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.97895
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.62786
14
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 5.82535
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 5.57754
15
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.8126
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.5424
16
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.51201
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.12141
17
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.46632
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.18582
18
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.94656
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.62783
19
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.84673
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.50759
20
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.77717
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.48694
21
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.60752
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.22237
```

```
22
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.00245
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.67395
23
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.18291
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.8332
24
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.43609
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.08212
25
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.17479
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.83852
26
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.11833
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.8353
27
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.5436
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 8.16732
28
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.34952
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.08038
29
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.71303
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.33804
30
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.5102
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.18206
31
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.54402
```

```
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.138
32
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.51466
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.16582
33
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.21418
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 5.94338
34
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.50835
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.14343
35
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.22165
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.89331
36
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.04631
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.68174
37
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.15415
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.76604
38
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.4013
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.04877
39
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.0146
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.70256
40
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.91757
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.56467
41
```

```
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.98573
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.66015
42
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.25933
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.94777
43
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.75716
Did not meet early stopping. Best iteration is:
[499]        valid_0's rmse: 6.47465
44
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.16996
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.78254
45
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.58973
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.24492
46
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.82723
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.36821
47
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.13485
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.66193
48
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.53662
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.16431
49
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.4559
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.12543
50
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.6723
Did not meet early stopping. Best iteration is:
```

```
[500]        valid_0's rmse: 7.31476
51
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.62583
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.30659
52
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.35062
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.93309
53
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.47556
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.19758
54
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.64458
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.34459
55
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.08038
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.6865
56
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.29881
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.90324
57
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.02756
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.61473
58
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.14933
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.77026
59
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.92266
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.49884
60
Training until validation scores don't improve for 200 rounds.
```

```
[400]        valid_0's rmse: 7.88688
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.52804
61
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.48576
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.06373
62
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.19272
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.82028
63
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.38449
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.02585
64
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.11569
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.71647
65
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.45324
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.0535
66
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.82172
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.40799
67
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.36472
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.98014
68
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.1103
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.71233
69
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.80035
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.33629
```

```
70
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.58045
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 8.08014
71
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.75235
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.36443
72
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.02358
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.64145
73
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.66678
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 8.23954
74
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.85491
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.41094
75
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.68435
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.2619
76
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.65893
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.23697
77
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.20564
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.80861
78
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.99465
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.64645
79
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.48784
```

```
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.13076
80
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.71216
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.2984
81
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.20194
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.7333
82
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.45732
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.0401
83
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.24123
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.80699
84
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.54319
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.09786
85
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.46638
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.06577
86
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.21256
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.7398
87
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.7743
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.36962
88
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.56759
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.13246
89
```

```
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.04759
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.59126
90
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.55653
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.11031
91
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.76199
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.27263
92
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.43478
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.02239
93
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.36077
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.90867
94
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.20647
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.71981
95
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.78353
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.32083
96
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.96664
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.35629
97
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.86067
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.34149
98
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.29375
Did not meet early stopping. Best iteration is:
```

```
[500]        valid_0's rmse: 6.84708
99
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.48115
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.86571
100
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.11178
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.57551
101
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.45643
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.89657
102
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 9.06178
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 8.47621
103
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.87323
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.36135
104
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.96335
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 8.31303
105
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.34483
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.7195
106
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 9.34205
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 8.71989
107
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.59241
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.99475
108
Training until validation scores don't improve for 200 rounds.
```

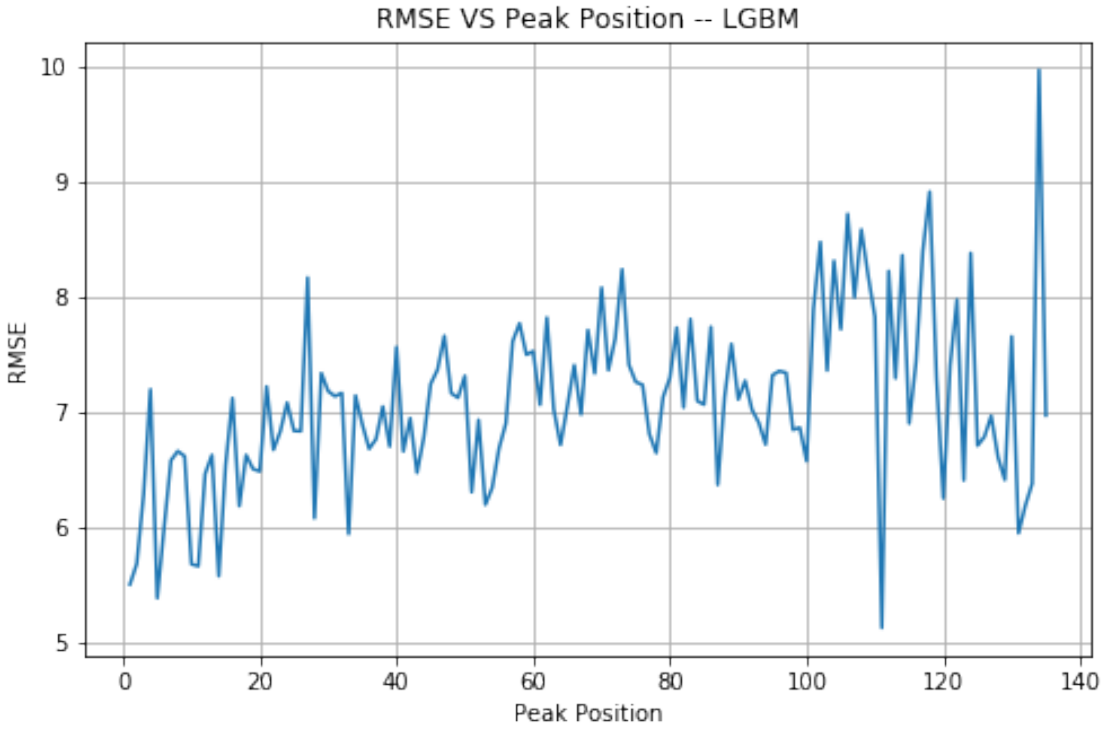
```
[400]        valid_0's rmse: 9.28763
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 8.58456
109
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.83183
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 8.18372
110
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.35595
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.82625
111
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 5.57295
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 5.12911
112
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.93566
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 8.22419
113
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.79017
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.29411
114
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 9.05258
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 8.36181
115
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.34583
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.90213
116
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.01235
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.43314
117
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 9.00586
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 8.39633
```

```
118
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 9.60515
Did not meet early stopping. Best iteration is:
[499]        valid_0's rmse: 8.91231
119
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.89953
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.29887
120
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.7703
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.25252
121
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.96431
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.40887
122
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.66407
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 7.97583
123
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 6.83299
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.4086
124
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 8.9766
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 8.38042
125
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.36678
Did not meet early stopping. Best iteration is:
[500]        valid_0's rmse: 6.71043
126
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.4013
Did not meet early stopping. Best iteration is:
[499]        valid_0's rmse: 6.78518
127
Training until validation scores don't improve for 200 rounds.
[400]        valid_0's rmse: 7.82766
```

```
Did not meet early stopping. Best iteration is:
[499]	valid_0's rmse: 6.96843
128
Training until validation scores don't improve for 200 rounds.
[400]	valid_0's rmse: 7.13613
Did not meet early stopping. Best iteration is:
[500]	valid_0's rmse: 6.60943
129
Training until validation scores don't improve for 200 rounds.
[400]	valid_0's rmse: 6.98728
Did not meet early stopping. Best iteration is:
[500]	valid_0's rmse: 6.41245
130
Training until validation scores don't improve for 200 rounds.
[400]	valid_0's rmse: 8.47427
Did not meet early stopping. Best iteration is:
[500]	valid_0's rmse: 7.65797
131
Training until validation scores don't improve for 200 rounds.
[400]	valid_0's rmse: 6.50553
Did not meet early stopping. Best iteration is:
[498]	valid_0's rmse: 5.94988
132
Training until validation scores don't improve for 200 rounds.
[400]	valid_0's rmse: 6.86245
Did not meet early stopping. Best iteration is:
[500]	valid_0's rmse: 6.1908
133
Training until validation scores don't improve for 200 rounds.
[400]	valid_0's rmse: 7.09447
Did not meet early stopping. Best iteration is:
[499]	valid_0's rmse: 6.38167
134
Training until validation scores don't improve for 200 rounds.
[400]	valid_0's rmse: 10.879
Did not meet early stopping. Best iteration is:
[500]	valid_0's rmse: 9.9695
135
Training until validation scores don't improve for 200 rounds.
[400]	valid_0's rmse: 7.65935
Did not meet early stopping. Best iteration is:
[500]	valid_0's rmse: 6.97215
```

```
In [86]: import matplotlib.pyplot as plt
         plt.figure(figsize=(8,5))
         plt.title("RMSE VS Peak Position -- LGBM")
         plt.xlabel("Peak Position")
```

```
plt.ylabel("RMSE")
plt.grid()
plt.plot(peak_list,rmse_list)
plt.show()
```



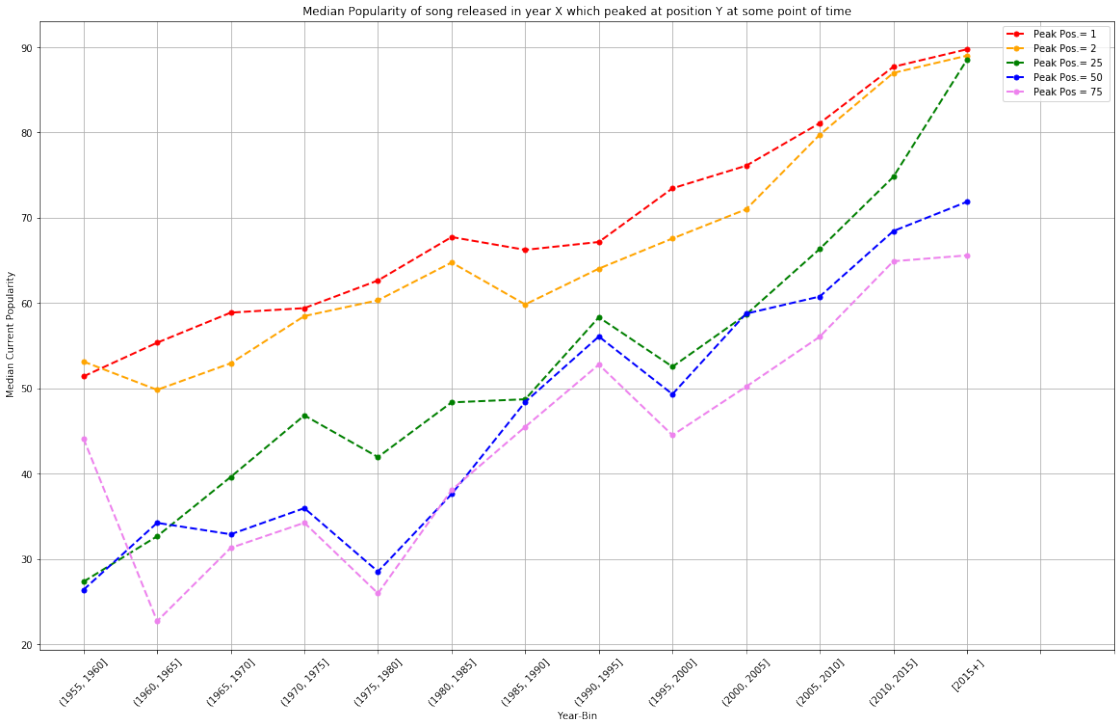

```
In [64]: df_peak1['year-bin'] = pd.cut(df_peak1['year'], bins = list(range(1950, 2020, 5))).as
         df_peak1.loc[df_peak1['year-bin'] == 'nan', 'year-bin'] = '[2015+]'

         df_peak2['year-bin'] = pd.cut(df_peak2['year'], bins = list(range(1950, 2020, 5))).as
         df_peak2.loc[df_peak2['year-bin'] == 'nan', 'year-bin'] = '[2015+]'

         df_peak25['year-bin'] = pd.cut(df_peak25['year'], bins = list(range(1950, 2020, 5))).
         df_peak25.loc[df_peak25['year-bin'] == 'nan', 'year-bin'] = '[2015+]'

         df_peak50['year-bin'] = pd.cut(df_peak50['year'], bins = list(range(1950, 2020, 5))).
         df_peak50.loc[df_peak50['year-bin'] == 'nan', 'year-bin'] = '[2015+]'

         df_peak75['year-bin'] = pd.cut(df_peak75['year'], bins = list(range(1950, 2020, 5))).
         df_peak75.loc[df_peak75['year-bin'] == 'nan', 'year-bin'] = '[2015+]'
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  """Entry point for launching an IPython kernel.
/opt/conda/lib/python3.6/site-packages/pandas/core/indexing.py:543: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[item] = s
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  after removing the cwd from sys.path.
/opt/conda/lib/python3.6/site-packages/pandas/core/indexing.py:543: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[item] = s
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  import sys
/opt/conda/lib/python3.6/site-packages/pandas/core/indexing.py:543: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[item] = s
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  # Remove the CWD from sys.path while we load stuff.
/opt/conda/lib/python3.6/site-packages/pandas/core/indexing.py:543: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[item] = s
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  del sys.path[0]
/opt/conda/lib/python3.6/site-packages/pandas/core/indexing.py:543: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  self.obj[item] = s
```

```
In [66]: import matplotlib.pyplot as plt
         plt.figure(figsize=(20,12))
         p1=df_peak1.groupby('year-bin')['net_current_popularity'].median().sort_index().plot(
         p2=df_peak2.groupby('year-bin')['net_current_popularity'].median().sort_index().plot(
         p3=df_peak25.groupby('year-bin')['net_current_popularity'].median().sort_index().plot
         p4=df_peak50.groupby('year-bin')['net_current_popularity'].median().sort_index().plot
         p5= df_peak75.groupby('year-bin')['net_current_popularity'].median().sort_index().plo
         # plt.legend((p1, p2, p3, p4),
         #            ('Peak Po.= 1', 'Peak Po.= 2', 'Peak Po.= 25', 'Peak Po.= 50'),
         #            fontsize=8)
         plt.gca().legend()
         plt.xlabel('Year-Bin')
         plt.ylabel(' Median Current Popularity')

         tick_labels = tuple(df_peak2['year-bin'].unique())
         x_max = int(max(plt.xticks()[0]))
         plt.xticks(range(0, x_max + 1), tick_labels, rotation=45)
         plt.grid()
         plt.title('Median Popularity of song released in year X which peaked at position Y at
         plt.show()
```



In [ ]:
```
# Top 5 overperforming songs in each period for songs that Peaked at 1 (super popular
df_overper = df_peak1.copy()
df_overper = df_overper.sort_values(by = ['year-bin','net_current_popularity'], ascendi
df_overper= df_overper.groupby('year-bin').head(5)
# df_overper = df_overper.sort_values(by = ['year-bin'], ascending = True)
```

In [ ]:
```
# Top 5 overperforming songs in each period:

df_under = df_peak1.copy()
df_under = df_under.sort_values(by = ['year-bin','net_current_popularity'], ascending =
df_under= df_under.groupby('year-bin').tail(5)
```

In [ ]:
```
df_median=df_peak1.groupby('year-bin').agg({'year-bin':'first','valence':'median','danc
df_median.head(15)
```

In [ ]:
```
# 1. Plot of how valence has changed among songs which peaked at #1 at some point base
# 2. Plot of the overperforming and underperforming songs in each 5 year bin
from matplotlib.pyplot import plot
plt.figure(figsize=(10,6))

x = df_overper['year-bin'].unique()
y = df_overper.groupby('year-bin')['valence'].median()
plt.scatter(x,y,c='red',alpha = 0.5,label = 'median overperforming')
```

```
x = df_under['year-bin'].unique()
y = df_under.groupby('year-bin')['valence'].median()
plt.scatter(x,y,c='blue',alpha = 0.5,label = 'median underperforming')
# df_peak1.groupby('year-bin')['valence'].median().sort_index().plot(color = 'green');
plt.plot(df_median['year-bin'], df_median['valence'], color='green', marker='o', lines
plt.title('Valence vs Year')

plt.gca().legend()
plt.xticks(rotation=45)
plt.ylabel('Median Valence')
plt.grid()
plt.show()
```

In [ ]:
```
# 1. Plot of how valence has changed among songs which peaked at #1 at some point base
# 2. Plot of the overperforming and underperforming songs in each 5 year bin
from matplotlib.pyplot import plot
plt.figure(figsize=(10,6))
y = df_overper.groupby('year-bin')['loudness'].median()
x = df_overper['year-bin'].unique()
plt.scatter(x,y,c='red',alpha = 0.5,label='median overperforming')

y = df_under.groupby('year-bin')['loudness'].median()
x = df_under['year-bin'].unique()
plt.scatter(x,y,c='blue',alpha = 0.5,label='median underperforming')

# df_peak1.groupby('year-bin')['valence'].median().sort_index().plot(color = 'green');
plot(df_median['year-bin'], df_median['loudness'], color='green', marker='o', linestyle
plt.title('Loudness vs Year')
plt.gca().legend()
plt.xticks(rotation=45)
plt.ylabel('Median Loudness')
plt.grid()

plt.show()
```

In [ ]:
```
from matplotlib.pyplot import plot
plt.figure(figsize=(10,6))
y = df_overper.groupby('year-bin')['danceability'].median()
x = df_overper['year-bin'].unique()
plt.scatter(x,y,c='red',alpha = 0.5,label='overperforming')

y = df_under.groupby('year-bin')['danceability'].median()
x = df_under['year-bin'].unique()
plt.scatter(x,y,c='blue',alpha = 0.5,label='underperforming')

# df_peak1.groupby('year-bin')['valence'].median().sort_index().plot(color = 'green');
plot(df_median['year-bin'], df_median['danceability'], color='green', marker='o', lines
plt.title('Danceability vs Year')
```

```
        plt.gca().legend()
        plt.xticks(rotation=45)
        plt.grid()
        plt.ylabel('Median Danceability')
        plt.show()
```

In [ ]:
```
from matplotlib.pyplot import plot
plt.figure(figsize=(10,6))
y = df_overper.groupby('year-bin')['acousticness'].median()
x = df_overper['year-bin'].unique()
plt.scatter(x,y,c='red',alpha = 0.5,label='overperforming')

y = df_under.groupby('year-bin')['acousticness'].median()
x = df_under['year-bin'].unique()
plt.scatter(x,y,c='violet',alpha = 0.5,label='underperforming')

# df_peak1.groupby('year-bin')['valence'].median().sort_index().plot(color = 'green');
plot(df_median['year-bin'], df_median['acousticness'], color='green', marker='o', lines
plt.title('acousticness vs Year')
plt.ylabel('Median Acousticness')
plt.gca().legend()
plt.xticks(rotation=45)
plt.grid()
plt.show()
```

In [ ]:
```
from matplotlib.pyplot import plot
plt.figure(figsize=(10,6))
y = df_overper.groupby('year-bin')['energy'].median()
x = df_overper['year-bin'].unique()
plt.scatter(x,y,c='red',alpha = 0.5,label='overperforming')

y = df_under.groupby('year-bin')['energy'].median()
x = df_under['year-bin'].unique()
plt.scatter(x,y,c='violet',alpha = 0.5,label='underperforming')

# df_peak1.groupby('year-bin')['valence'].median().sort_index().plot(color = 'green');
plot(df_median['year-bin'], df_median['energy'], color='green', marker='o', linestyle=
plt.title('energy vs Year')
plt.gca().legend()
plt.xticks(rotation=45)
plt.grid()
plt.show()
```

In [ ]:
```
## 2nd Possible Approach
# Top 5 overperforming  --> top 10 songs which peaked at rank 100
# Top 5 underperforming --> bottom 10 songs which peaked at rank 1
```

In [ ]:
```
import matplotlib.pyplot  as plt
```

```
df_med_op=df_peak50.groupby('year-bin').agg({'year-bin':'first','valence':'median','dar
df_med_ud=df_peak1.groupby('year-bin').agg({'year-bin':'first','valence':'median','danc

# plt.subplot(2, 1, 1)
plt.figure(figsize=(10,6))
df_op = df_peak50.copy()
df_op = df_op.sort_values(by = ['year-bin','net_current_popularity'], ascending = [True
df_op= df_op.groupby('year-bin').head(5)
plt.plot(df_med_op['year-bin'], df_med_op['danceability'], color='green', marker='o', l
y = df_op.groupby('year-bin')['danceability'].median()
x = df_op['year-bin'].unique()
plt.scatter(x,y,c='red',alpha = 0.5)
plt.title('Danceability vs Year ')
# plt.show()

# plt.subplot(2, 1, 2)
# plt.figure(figsize=(18,5))
df_ud = df_peak1.copy()
df_ud = df_ud.sort_values(by = ['year-bin','net_current_popularity'], ascending = [True
df_ud= df_ud.groupby('year-bin').tail(5)
plt.plot(df_med_ud['year-bin'], df_med_ud['danceability'], color='green', marker='o', l
y = df_ud.groupby('year-bin')['danceability'].median()
x = df_ud['year-bin'].unique()
plt.scatter(x,y,c='violet',alpha = 0.5)
# plt.title('Danceability vs Year underperforming')
plt.gca().legend()
plt.xticks(rotation=45)
plt.grid()
plt.show()
```

In [ ]:
```
import matplotlib.pyplot  as plt

df_med_op=df_peak50.groupby('year-bin').agg({'year-bin':'first','valence':'median','dar
df_med_ud=df_peak1.groupby('year-bin').agg({'year-bin':'first','valence':'median','danc

# plt.subplot(2, 1, 1)
plt.figure(figsize=(18,5))
df_op = df_peak50.copy()
df_op = df_op.sort_values(by = ['year-bin','net_current_popularity'], ascending = [True
df_op= df_op.groupby('year-bin').head(5)
# plt.plot(df_med_op['year-bin'], df_med_op['loudness'], color='green', marker='o', li
y = df_op.groupby('year-bin')['loudness'].median()
x = df_op['year-bin'].unique()
plt.scatter(x,y,c='red',alpha = 0.5)
# plt.title('Loudness vs Year overperforming')
# plt.show()

# plt.subplot(2, 1, 2)
```

```
        # plt.figure(figsize=(18,5))
        df_ud = df_peak1.copy()
        df_ud = df_ud.sort_values(by = ['year-bin','net_current_popularity'], ascending = [Tru
        df_ud= df_ud.groupby('year-bin').tail(5)
        plt.plot(df_med_ud['year-bin'], df_med_ud['loudness'], color='green', marker='o', lines
        y = df_ud.groupby('year-bin')['loudness'].median()
        x = df_ud['year-bin'].unique()
        plt.scatter(x,y,c='violet',alpha = 0.5)
        plt.title('Loudness vs Year ')
        plt.show()

In [ ]: # Finding Artists With Song Popularity
        # Artists -> The Beatles
        df_artist = df_base.sort_values(by ='net_current_popularity', ascending = False)
        df_artist = df_artist[df_artist['Artist_Name'].str.contains("SINATRA")]
        df_artist = df_artist.sort_values(by ='year', ascending = False)
        df_artist.head(100)

In [ ]: # Frank Sinatra
        import matplotlib.pyplot as plt
        df_artist = df_base.sort_values(by ='net_current_popularity', ascending = False)
        df_artist = df_artist[df_artist['Artist_Name'].str.contains("SINATRA")]
        df_artist = df_artist.sort_values(by ='year', ascending = False)
        df_artist = df_artist[df_artist['year']<1975]
        df_artist.head(100)

        plt.figure(figsize=(10,5))
        x = df_artist['year']
        y = df_artist['net_current_popularity']
        plt.xlabel('Year')
        plt.ylabel("Current Popularity")
        plt.grid()
        plt.scatter(x,y,color='red',alpha=0.5)

In [ ]: # Frank Sinatra
        df_artist = df_base.sort_values(by ='net_current_popularity', ascending = False)
        df_artist = df_artist[df_artist['Artist_Name'].str.contains("BEATLES")]
        df_artist = df_artist[df_artist['year']<1975]
        df_artist = df_artist.sort_values(by ='year', ascending = False)
        df_artist.head(100)

        plt.figure(figsize=(10,8))
        x = df_artist['year']
        y = df_artist['net_current_popularity']
        plt.xlabel('Year')
        plt.ylabel("Current Popularity")
        plt.grid()
        plt.scatter(x,y,color='red',alpha=0.5)
```

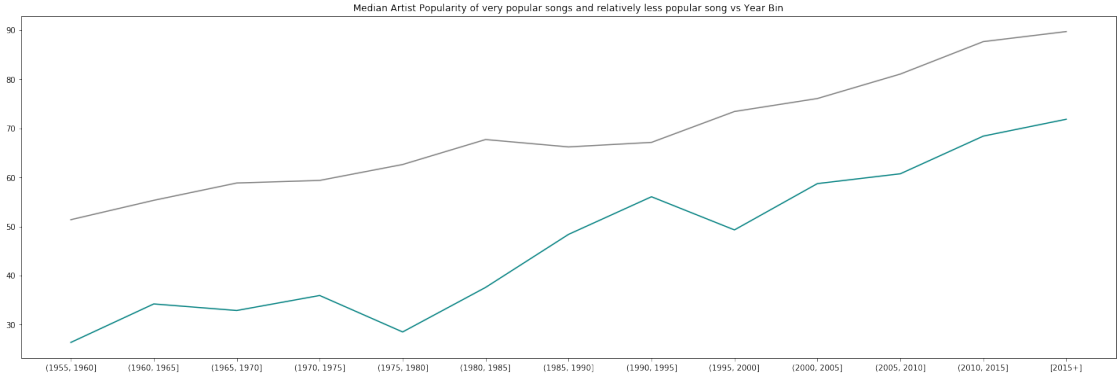
```
In [ ]: # Frank Sinatra
        df_artist = df_base.sort_values(by ='net_current_popularity', ascending = False)
        df_artist = df_artist[df_artist['Artist_Name'].str.contains("EAGLES")]
        df_artist = df_artist.sort_values(by ='year', ascending = False)
        df_artist.head(100)

        plt.figure(figsize=(10,4))
        x = df_artist['year']
        y = df_artist['net_current_popularity']
        plt.scatter(x,y)

In [ ]: # Frank Sinatra
        df_artist = df_base.sort_values(by ='net_current_popularity', ascending = False)
        df_artist = df_artist[df_artist['Artist_Name'].str.contains("ROLLING STONES")]
        df_artist = df_artist.sort_values(by ='year', ascending = False)
        df_artist.head(100)

        plt.figure(figsize=(10,6))
        x = df_artist['year']
        y = df_artist['net_current_popularity']
        plt.grid()
        plt.scatter(x,y,color='red',alpha=0.5)

In [ ]: # Frank Sinatra
        df_artist = df_base.sort_values(by ='net_current_popularity', ascending = False)
        df_artist = df_artist[df_artist['Artist_Name'].str.contains("ELVIS PRESLEY")]
        df_artist = df_artist.sort_values(by ='year', ascending = False)
        df_artist = df_artist[df_artist['year']<2000]
        df_artist.head(100)

        plt.figure(figsize=(10,6))
        x = df_artist['year']
        plt.xlabel('Year')
        plt.ylabel("Current Popularity")

        y = df_artist['net_current_popularity']
        plt.grid()
        plt.scatter(x,y,color='red',alpha=0.5)

In [ ]: # Eagles
        df_artist = df_base.sort_values(by ='net_current_popularity', ascending = False)
        df_artist = df_artist[df_artist['Artist_Name'].str.contains("ELTON JOHN")]
        df_artist = df_artist.sort_values(by ='year', ascending = False)
        df_artist.head(100)

        plt.figure(figsize=(10,4))
        x = df_artist['year']
        y = df_artist['net_current_popularity']
```

```
        plt.grid()
        plt.ylabel('Current Popularity')
        plt.xlabel('Year')
        plt.scatter(x,y,c='red',alpha = 0.5)

In [ ]: # Frank Sinatra
        df_artist = df_base.sort_values(by ='net_current_popularity', ascending = False)
        df_artist = df_artist[df_artist['Artist_Name'].str.contains("BOB DYLAN")]
        df_artist = df_artist.sort_values(by ='year', ascending = False)
        df_artist.head(100)

        plt.figure(figsize=(1,4))
        x = df_artist['year'].unique()
        y = df_artist.groupby('year')['net_current_popularity'].median()
        plt.bar(x,y)

In [ ]: df_topmosthit.head(20)

In [ ]: #One hit wonders

        df_onehit = df_base.copy()
        df_onehit['artist_count']=df_onehit.groupby('Artist')['Artist'].transform('count')

        df_onehit=df_onehit.sort_values(by =['year','artist_count','net_current_popularity'], a
        df_topmosthit = df_onehit.groupby('year').head(1)

In [ ]: import seaborn as sns
        ax = sns.barplot(x=df_topmosthit['year'], y="total_bill", hue="sex", data=tips)

In [ ]: plt.figure(figsize=(15,8))

        x = df_topmosthit['year']
        y = df_topmosthit['net_current_popularity']
        plt.scatter(x,y,c='red',alpha = 0.5,label='One Hit Wonder')

        df_mostpop = df_base.copy()
        df_mostpop['artist_count']=df_mostpop.groupby('Artist')['Artist'].transform('count')
        df_mostpop = df_mostpop.sort_values(by =['year','net_current_popularity'], ascending=
        df_maxpop = df_mostpop.groupby('year').head(1)
        x = df_maxpop['year']
        y = df_maxpop['net_current_popularity']
        plt.grid()
        plt.ylabel('Current Popularity')
        plt.scatter(x,y,c='blue',alpha = 0.5,label = 'Most Popular Song')
        plt.gca().legend()
        plt.show()

In [ ]: df_topmosthit.head(20)
```

```
In [ ]: df_mrg = pd.merge(df_topmosthit, df_maxpop, on="year")

        df_mrg= df_mrg[df_mrg['net_current_popularity_y']-df_mrg['net_current_popularity_x'] <
        df_mrg
        df_mrg.head(10)
```

1970 --> In the summertime 1994. --> Juicy 1995 --> Gangsta's Paradise

```
In [ ]: df_topmosthit[df_topmosthit['year']==2000]

In [ ]: df_maxpop[df_maxpop['year']==2000]

In [ ]: import matplotlib.pyplot  as plt
        plt.figure(figsize=(30,7))
        df_artistpop = df_base.copy()
        df_artistpop.sort_values(by='artist popularity',ascending=False)
        x=df_artistpop['artist popularity'].unique()
        y=df_artistpop.groupby('artist popularity')['net_current_popularity'].median()
        plt.xlabel('Artist popularity')
        plt.ylabel('Song popularity')
        plt.title('Artist Popularity vs Current Song Popularity')
        plt.bar(x,y)

In [ ]: df_base.groupby('')

In [87]: plt.figure(figsize=(25,8))
         df_peak = df_peak1.sort_values(by = ['year-bin','net_current_popularity'], ascending =
         df_peak_median1 = df_peak.copy()
         df_peak_median1['median_pop1'] = df_peak_median1.groupby('year-bin')['net_current_popu
         # df_peak_median1[' median_pop1'] = df_peak_median1.groupby('year-bin').agg({'year-bi
         plt.plot(df_peak_median1['year-bin'], df_peak_median1['median_pop1'], color='grey')

         df_peak = df_peak50.sort_values(by = ['year-bin','net_current_popularity'], ascending
         df_peak_median2 = df_peak.copy()
         df_peak_median2['median_pop2'] = df_peak_median2.groupby('year-bin')['net_current_popu
         # df_peak_median2['median_pop2'] = df_peak_median2.groupby('year-bin').agg({'year-bin
         plt.plot(df_peak_median2['year-bin'], df_peak_median2['median_pop2'], color='teal')

         # plt.bar(df_peak_median1['year-bin'], df_peak_median1['median_pop1'], align='center'
         # plt.bar(df_peak_median2['year-bin'], df_peak_median2['median_pop2'], align='center'

         # df_plot = df_peak_median1.copy()
         # df_plot['median_pop2'] = df_peak_median2['median_pop2']
         # df_plot.plot(x='year-bin',y=['median_pop1','median_pop2'],kind='bar',figsize=(30,40
         plt.title('Median Artist Popularity of very popular songs and relatively less popular
         #df_plot.plot(kind='bar',figsize=(20,10))
         plt.show()
```




```
In [ ]: df_sniff=df_base.head()
```