Modeling - Clean

In [1]:
```
import io
import surprise
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
from surprise import Reader, Dataset
from surprise.prediction_algorithms import SVD, SVDpp, BaselineOnly, KNNWit
from surprise.prediction_algorithms import NMF, SlopeOne, NormalPredictor
from surprise.model_selection import GridSearchCV, cross_validate, train_te
```

In [2]:
```
full = pd.read_csv('../Data/filtered-cleaned')
full = full.drop(columns = 'Unnamed: 0')
min_cols = full[[ 'userId', 'movieId', 'rating']]
min_cols = min_cols.sample(500000)
smaller = min_cols.sample(50000)
```

In [3]:
```
reader = Reader()
data = Dataset.load_from_df(min_cols, reader)
datasmall = Dataset.load_from_df(smaller, reader)
kdata = datasmall.build_full_trainset()
trainset, testset = train_test_split(data, test_size = 0.10)
```

In [4]:
```
BaselineOnly_results = cross_validate(BaselineOnly(), datasmall, verbose =
```

```
Estimating biases using als...
Estimating biases using als...
Estimating biases using als...
Estimating biases using als...
Estimating biases using als...
```

In [5]:
```
nmf_results = cross_validate(NMF(), datasmall)
```

In [6]:
```
NormalPredictor_results = cross_validate(NormalPredictor(), datasmall)
```

In [7]:
```
SlopeOne_results = cross_validate(SlopeOne(), datasmall)
```

In [8]:
```
SVDpp_results = cross_validate(SVDpp(), datasmall)
```

In [9]:
```
SVD_results = cross_validate(SVD(), datasmall)
```

In [10]:
```
results_list = [SVD_results, SVDpp_results, SlopeOne_results,
                NormalPredictor_results, nmf_results, BaselineOnly_results]
```

In [11]:
```
def get_metrics(lst_dicts, key1, key2):
    values = []
    for dct in lst_dicts:
        values.append([dct[key1], dct[key2]])
    return pd.DataFrame(values, columns=[key1, key2])
```

In [12]:
```
metrics_df = get_metrics(results_list, 'test_mae', 'test_rmse')
```

In [13]:
```
result_names = pd.Series(['SVD_results', 'SVDpp_results', 'SlopeOne_results
                          'NormalPredictor_results', 'nmf_results', 'Baseli
```

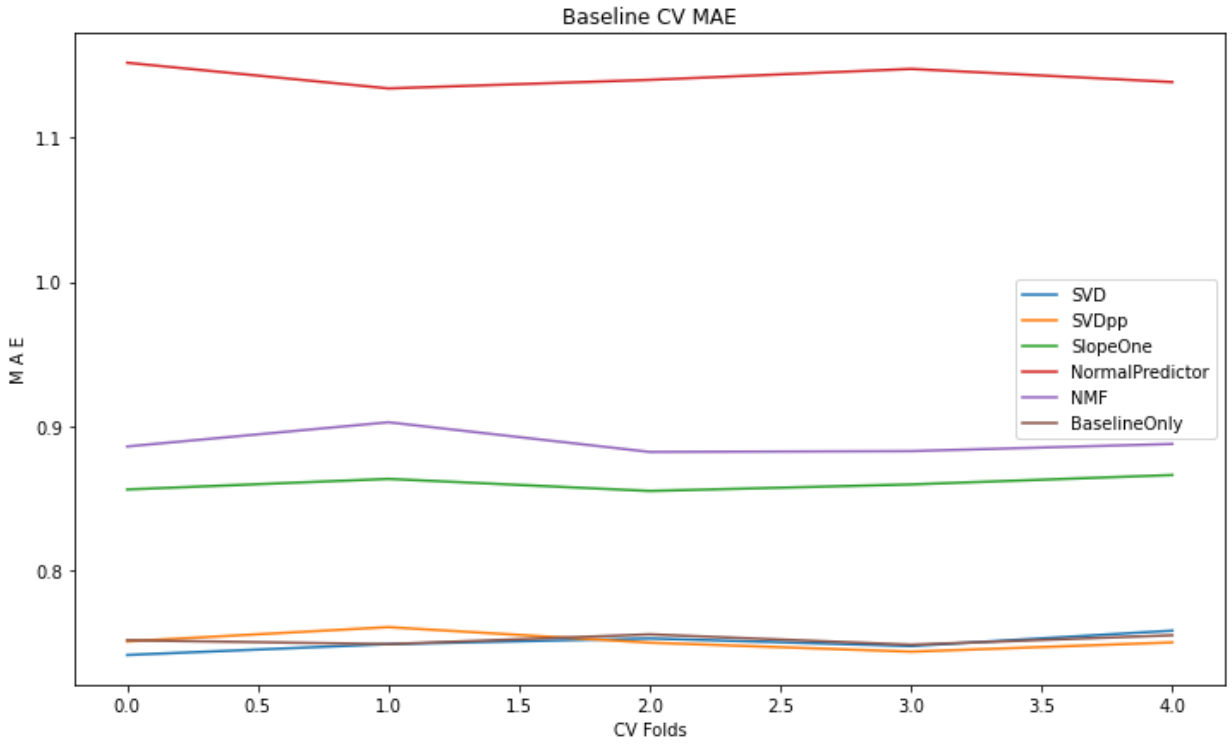
In [14]:
```
sum_df = metrics_df.merge(result_names.rename('models'), left_index = True,
```

In [15]:
```
sum_df
```

Out[15]:

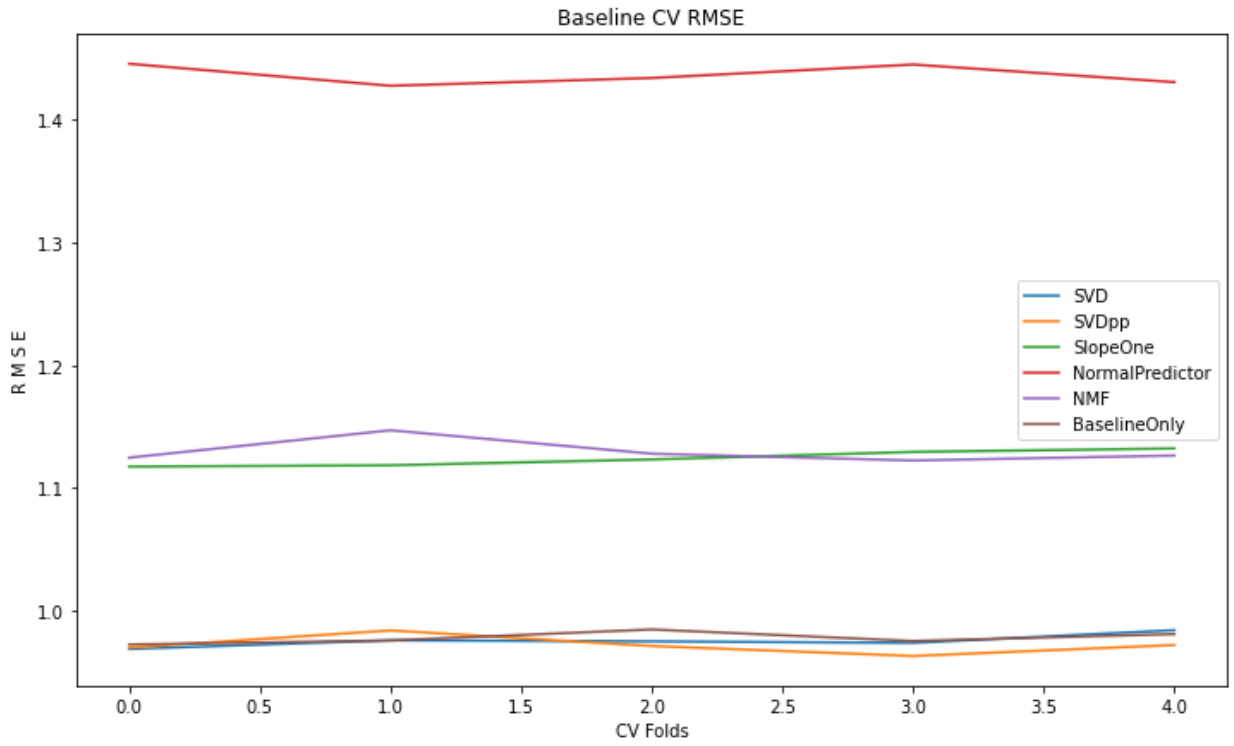
	test_mae	test_rmse	models
0	[0.7409482071867035, 0.7484976648417914, 0.752...	[0.9693487319897914, 0.9763622248891681, 0.975...	SVD_results
1	[0.7503864194529065, 0.7602743853639213, 0.749...	[0.9704501729966705, 0.984427119961496, 0.9717...	SVDpp_results
2	[0.8558602186192317, 0.8632069779080807, 0.854...	[1.1175612749857284, 1.118612796469587, 1.1233...	SlopeOne_results
3	[1.152124112846978, 1.1343031597434767, 1.1402...	[1.444876623357516, 1.4269764197105033, 1.4333...	NormalPredictor_results
4	[0.8857151987480205, 0.9025205683314862, 0.881...	[1.1248128338851642, 1.1470412718681522, 1.128...	nmf_results
5	[0.7512022316085034, 0.7484868989319579, 0.755...	[0.9729986061841047, 0.9762757925192751, 0.985...	BaselineOnly_results

In [16]:
```
plt.figure(figsize= (12, 7))
plt.plot(sum_df['test_mae'][0])
plt.plot(sum_df['test_mae'][1])
plt.plot(sum_df['test_mae'][2])
plt.plot(sum_df['test_mae'][3])
plt.plot(sum_df['test_mae'][4])
plt.plot(sum_df['test_mae'][5])
plt.title('Baseline CV MAE')
plt.xlabel('CV Folds')
plt.ylabel('M A E')
plt.legend(['SVD', 'SVDpp', 'SlopeOne', 'NormalPredictor', 'NMF', 'Baseline
plt.show()
```



In [17]:
```
plt.figure(figsize= (12, 7))
plt.plot(sum_df['test_rmse'][0])
plt.plot(sum_df['test_rmse'][1])
plt.plot(sum_df['test_rmse'][2])
plt.plot(sum_df['test_rmse'][3])
plt.plot(sum_df['test_rmse'][4])
plt.plot(sum_df['test_rmse'][5])
plt.title('Baseline CV RMSE')
plt.xlabel('CV Folds')
plt.ylabel('R M S E')

plt.legend(['SVD', 'SVDpp', 'SlopeOne', 'NormalPredictor', 'NMF', 'Baseline
plt.show()
```



In [18]:
```
svd_best = SVD(n_epochs = 50, n_factors = 75, reg_all = 0.2)
svd_best.fit(trainset)
```

Out[18]:
```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7fb8aa59167
0>
```

In [19]:
```
test_mae = surprise.accuracy.mae(svd_best.test(testset))
test_mae
```

```
MAE:  0.6756
```

Out[19]:
```
0.6755617246525298
```

In [20]:
```
predictions  = svd_best.test(testset)
```

In [21]:
```
test_list = []
for i in testset:
  test_list.append(i[2])
prediction_list = []
for i in predictions:
  prediction_list.append(i[3])
correlation = pearsonr(prediction_list, test_list)
correlation
```

Out[21]:
```
(0.5364997422005119, 0.0)
```