APPENDIX
Python Codes

```
# Importing the data
train = pd.read_csv("movielens_100k.base", sep="\t", header=None,
                names=["user_id", "movie_id", "rating", "timestamp"])
test = pd.read_csv("movielens_100k.test", sep="\t", header=None,
                names=["user_id", "movie_id", "rating", "timestamp"])
train.shape
test.shape

# Exploratory Data Analysis
train.describe()
test.describe()

#get bar plot for raitings in train data
df1 = pd.DataFrame(train["rating"])
df1

category_counts1 = df1['rating'].value_counts()
category_counts1

# Train Data: Rate Values
labels1= ['1', '2', '3', '4','5']
values1 = [4719, 9178, 21963, 27396,16744]

total1= sum(values1)
percentages1= [(value1 / total1) * 100 for value1 in values1]

fig, ax = plt.subplots()
ax.bar(labels1, percentages1, color=["black", "blue", "red", "green", "purple"])

for i, v in enumerate(percentages1):
    ax.text(i, v + 1, str(round(v, 1)) + '%', ha='center')

# Set chart title and axis labels
ax.set_title('Bar Plot of Ratings')
ax.set_xlabel('Raiting')
ax.set_ylabel('Percentages')

#get bar plot for raitings in test data
df2 = pd.DataFrame(test["rating"])
df2

category_counts2 = df2['rating'].value_counts()
category_counts2

# Train Data: Rate Values
labels2= ['1', '2', '3', '4','5']
values2 = [1391, 2192, 4457, 5182, 6778]

total2 = sum(values2)
percentages2 = [(value2 / total2) * 100 for value2 in values2]

fig, ax = plt.subplots()
ax.bar(labels2, percentages2, color=["black", "blue", "red", "green", "purple"])
```

```
for i, v in enumerate(percentages2):
    ax.text(i, v + 1, str(round(v, 1)) + '%', ha='center')
ax.set_title('Bar Plot of Ratings')
ax.set_xlabel('Raiting')
ax.set_ylabel('Percentages')

# Converting train data to CSR Matrix
users = train["user_id"].unique()
movies = train["movie_id"].unique()
shape = (len(users), len(movies))

# Create indices for users and movies
user_cat = CategoricalDtype(categories=sorted(users), ordered=True)
movie_cat = CategoricalDtype(categories=sorted(movies), ordered=True)
user_index = train["user_id"].astype(user_cat).cat.codes
movie_index = train["movie_id"].astype(movie_cat).cat.codes

# Conversion via COO matrix
coo = sparse.coo_matrix((train["rating"], (user_index, movie_index)), shape=shape)
train_data = coo.tocsr()
train_data2 = coo.todense()

def mat_factorization_func(R, P, Q, K, steps=5000, alpha=0.0002, beta=0.02):
    Q = Q.T
    for step in range(steps):
        for i in range(len(R)):
            for j in range(len(R[i])):
                if R[i][j] > 0:
                    eij = R[i][j] - np.dot(P[i,:],Q[:,j])
                    for k in range(K):
                        P[i][k] = P[i][k] + alpha * (2 * eij * Q[k][j] - beta * P[i][k])
                        Q[k][j] = Q[k][j] + alpha * (2 * eij * P[i][k] - beta * Q[k][j])
        #eR = np.dot(P,Q)
        e = 0
        for i in range(len(R)):
            for j in range(len(R[i])):
                if R[i][j] > 0:
                    e = e + pow(R[i][j] - np.dot(P[i,:],Q[:,j]), 2)
                    for k in range(K):
                        e = e + (beta/2) * (pow(P[i][k],2) + pow(Q[k][j],2))
        if e < 0.001:
            break
    return P, Q.T

train_data = np.array(train_data2)

np.random.seed(6785307)

N = len(train_data)
M = len(train_data[0])
K = 2
P = np.random.rand(N, K)
Q = np.random.rand(M, K)
nP, nQ = mat_factorization_func(train_data, P, Q, K)
nP
nQ
```

```
R2 = np.dot(nP, nQ.T)
R2

pos_in_train = train_data[np.nonzero(train_data)]
pos_out_train = R2[np.nonzero(train_data)]

MSE_train = np.mean(pow((pos_in_train - pos_out_train), 2))
RMSE_train = np.sqrt(MSE_train)
round(RMSE_train, 4)

# Converting test data to CSR Matrix
users2 = test["user_id"].unique()
movies2 = test["movie_id"].unique()
shape2 = (len(users2), len(movies2))

# Create indices for users and movies
user_cat2 = CategoricalDtype(categories=sorted(users2), ordered=True)
movie_cat2 = CategoricalDtype(categories=sorted(movies2), ordered=True)
user_index2 = test["user_id"].astype(user_cat2).cat.codes
movie_index2 = test["movie_id"].astype(movie_cat2).cat.codes

# Conversion via COO matrix
coo2 = sparse.coo_matrix((test["rating"], (user_index2, movie_index2)), shape=shape2)
#test_data = coo2.tocsr()
test_data2 = coo2.todense()
test_data = np.array(test_data2)

pos_in_test = test_data[np.nonzero(test_data)]
pos_out_test = R2[np.nonzero(test_data)]
len(pos_in_test)
len(pos_out_test)

MSE_test = np.mean(pow((pos_in_test - pos_out_test), 2))
RMSE_test = np.sqrt(MSE_test)
round(RMSE_test, 4)

#====================== Running witk k=3
N1 = len(train_data)
M1 = len(train_data[0])
K1 = 3
P1 = np.random.rand(N1, K1)
Q1 = np.random.rand(M1, K1)
nP1, nQ1 = mat_factorization_func(train_data, P1, Q1, K1)

R22 = np.dot(nP1, nQ1.T)
R22

# Train RMSE
pos_in_train2 = train_data[np.nonzero(train_data)]
pos_out_train2 = R22[np.nonzero(train_data)]

MSE_train2 = np.mean(pow((pos_in_train2 - pos_out_train2), 2))
RMSE_train2 = np.sqrt(MSE_train2)
round(RMSE_train2, 4)
```