# SGD Algorithm to predict movie ratings

**There will be some functions that start with the word "grader" ex: grader_matrix(), grader_mean(), grader_dim() etc, you should not change those function definition.**

**Every Grader function has to return True.**

1. Download the data from here
2. The data will be of this format, each data point is represented as a triplet of user_id, movie_id and rating

| user_id | movie_id | rating |
|---|---|---|
| 77 | 236 | 3 |
| 471 | 208 | 5 |
| 641 | 401 | 4 |
| 31 | 298 | 4 |
| 58 | 504 | 5 |
| 235 | 727 | 5 |

## Task 1

### Predict the rating for a given (user_id, movie_id) pair

Predicted rating $\hat{y}_{ij}$ for user i, movied j pair is calcuated as $\hat{y}_{ij} = \mu + b_i + c_j + u_i^T v_j$ , here we will be finding the best values of $b_{i}$ and $c_{j}$ using SGD algorithm with the optimization problem for N users and M movies is defined as

$$ L = \min_{ b, c, \{ u_i \}_{i=1}^N, \{ v_j \}_{j=1}^M} \quad \alpha \Big( \sum_{j} \sum_{k} v_{jk}^2 + \sum_{i} \sum_{k} u_{ik}^2 + \sum_{i} b_i^2 + \sum_{j} c_i^2 \Big) + \sum_{i,j \in \mathcal{I}^{\text{train}}} (y_{ij} - \mu - b_i - c_j - u_i^T v_j)^2 $$

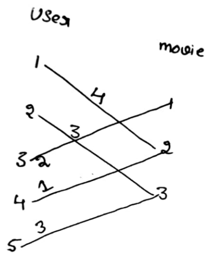
- $\mu$ : scalar mean rating
- $b_i$ : scalar bias term for user $i$
- $c_j$ : scalar bias term for movie $j$
- $u_i$ : K-dimensional vector for user $i$
- $v_j$ : K-dimensional vector for movie $j$

$ \ $

*. We will be giving you some functions, please write code in that functions only.

*. After every function, we will be giving you expected output, please make sure that you get that output.

1. Construct adjacency matrix with the given data, assuming its weighted un-directed bi-partited graph and the weight of each edge is the rating given by user to the movie




the Adjacency matrix

$$\begin{array}{c|c}
 & \begin{matrix} 1 & 2 & 3 \end{matrix} \\
\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 4 & 0 \\ 0 & 0 & 3 \\ 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}
\end{array}$$

you can construct this matrix like $A[i][j]=r_{ij}$ here $i$ is user_id, $j$ is movie *id and $r*{ij}$ is rating given by user $i$ to the movie $j$

Hint : you can create adjacency matrix using csr_matrix

1. We will Apply SVD decomposition on the Adjaceny matrix link1, link2 and get three matrices $U, \sum, V$ such that $U \times \sum \times V^T = A$,
   if $A$ is of dimensions $N \times M$ then
   U is of $N \times k$,
   $\sum$ is of $k \times k$ and
   $V$ is $M \times k$ dimensions.

   *. So the matrix $U$ can be represented as matrix representation of users, where each row $u_{i}$ represents a k-dimensional vector for a user

   *. So the matrix $V$ can be represented as matrix representation of movies, where each row $v_{j}$ represents a k-dimensional vector for a movie.

2. Compute $\mu$ , $\mu$ represents the mean of all the rating given in the dataset.(write your code in def m_u())
3. For each unique user initilize a bias value $B_{i}$ to zero, so if we have $N$ users $B$ will be a $N$ dimensional vector, the $i^{th}$ value of the $B$ will corresponds to the bias term for $i^{th}$ user (write your code in def initialize())

4. For each unique movie initilize a bias value $C_{j}$ zero, so if we have $M$ movies $C$ will be a $M$ dimensional vector, the $j^{th}$ value of the $C$ will corresponds to the bias term for $j^{th}$ movie (write your code in def initialize())

5. Compute dL/db_i (Write you code in def derivative_db())

6. Compute dL/dc_j(write your code in def derivative_dc()

7. Print the mean squared error with predicted ratings.

```
for each epoch:
    for each pair of (user, movie):
        b_i =  b_i - learning_rate * dL/db_i
        c_j =  c_j - learning_rate * dL/dc_j
predict the ratings with formula
```

$\hat{y}_{ij} = \mu + b_i + c_j + \text{dot_product}(u_i , v_j) $

1. you can choose any learning rate and regularization term in the range $10^{-3} \text{ to } 10^2$

2. **bonus**: instead of using SVD decomposition you can learn the vectors $u_i$, $v_j$ with the help of SGD algo similar to $b_i$ and $c_j$

# Task 2

As we know U is the learned matrix of user vectors, with its i-th row as the vector ui for user i. Each row of U can be seen as a "feature vector" for a particular user.

The question we'd like to investigate is this: do our computed per-user features that are optimized for predicting movie ratings contain anything to do with gender?

The provided data file user_info.csv contains an is_male column indicating which users in the dataset are male. Can you predict this signal given the features U?

> **Note 1** : there is no train test split in the data, the goal of this assignment is to give an intution about how to do matrix factorization with the help of SGD and application of truncated SVD. for better understanding of the collabarative fillerting please check netflix case study.
>
> **Note 2** : Check if scaling of $U$, $V$ matrices improve the metric

Reading the csv file

In [1]:

```
import pandas as pd
data=pd.read_csv('ratings_train.csv')
data.head()
```

Out[1]:

| | user_id | item_id | rating |
|---|---|---|---|
| **0** | 772 | 36 | 3 |
| **1** | 471 | 228 | 5 |
| **2** | 641 | 401 | 4 |
| **3** | 312 | 98 | 4 |
| **4** | 58 | 504 | 5 |

In [2]:

```
data.shape
```

Out[2]:

```
(89992, 3)
```

Create your adjacency matrix

In [3]:

```
from scipy.sparse import csr_matrix
row = data['user_id'].values
col = data['item_id'].values
value = data['rating'].values
adjacency_matrix = csr_matrix((value, (row, col)))
```

In [4]:

```
adjacency_matrix.shape
```

Out[4]:

```
(943, 1681)
```

Grader function - 1

In [5]:

```
def grader_matrix(matrix):
  assert(matrix.shape==(943,1681))
  return True
grader_matrix(adjacency_matrix)
```

Out[5]:

```
True
```

SVD decompostion

Sample code for SVD decompostion

In [6]:

```
from sklearn.utils.extmath import randomized_svd
import numpy as np
matrix = np.random.random((20, 10))
U, Sigma, VT = randomized_svd(matrix, n_components=5,n_iter=5, random_state=None)
print(U.shape)
print(Sigma.shape)
print(VT.T.shape)
```

```
(20, 5)
(5,)
(10, 5)
```

Write your code for SVD decompostion

In [29]:

```
U, Sigma_rating, VT = randomized_svd(adjacency_matrix, n_components=500,n_iter=10, random_state=0)
print(U.shape)
print(Sigma_rating.shape)
print(VT.T.shape)
```

```
(943, 500)
(500,)
(1681, 500)
```

In [55]:

```
VT.shape
```

Out[55]:

```
(500, 1681)
```

Compute mean of ratings

In [9]:

```
def m_u(ratings):
    '''In this function, we will compute mean for all the ratings'''
    # you can use mean() function to do this
    # check this (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.mean.html)
    mean = ratings.mean()

    return mean
```

In [10]:

```
mu=m_u(data['rating'])
print(mu)
```

```
3.529480398257623
```

Grader function -2

In [13]:

```
def grader_mean(mu):
  assert(np.round(mu,3)==3.529)
  return True
mu=m_u(data['rating'])
grader_mean(mu)
```

Out[13]:

```
True
```

Initialize $B_{i}$ and $C_{j}$

Hint : Number of rows of adjacent matrix corresponds to user dimensions($B_{i}$), number of columns of adjacent matrix corresponds to movie dimensions ($C_{j}$)

In [15]:

```
def initialize(dim):
    '''In this function, we will initialize bias value 'B' and 'C'.'''
    # initalize the value to zeros
    # return output as a list of zeros
    vec = np.zeros(dim)
    return vec
```

In [16]:

```
dim= U.shape[0]# give the number of dimensions for b_i (Here b_i corresponds to users)
b_i=initialize(dim)
```

In [17]:

```
dim= M.shape[0]# give the number of dimensions for c_j (Here c_j corresponds to movies)
c_j=initialize(dim)
```

Grader function -3

In [18]:

```
def grader_dim(b_i,c_j):
  assert(len(b_i)==943 and np.sum(b_i)==0)
  assert(len(c_j)==1681 and np.sum(c_j)==0)
  return True
grader_dim(b_i,c_j)
```

Out[18]:

```
True
```

Compute dL/db_i

In [68]:

```
def derivative_db(user_id,item_id,rating,U,V,mu,alpha):
    '''In this function, we will compute dL/db_i'''
    bi = b_i[user_id]
    bi_2 = bi ** 2
    u = U[user_id]
    v = V[:,item_id]
    cj = c_j[item_id]
    cj_2 = cj ** 2
    y_ij = adjacency_matrix[user_id, item_id]

    value = (2 * alpha * bi * (np.dot(v,v) + np.dot(u,u) + bi_2 + cj_2)) - (2 * (y_ij - mu - bi - cj - (np.
    return value
```

Grader function -4

In [69]:

```
def grader_db(value):
    assert(np.round(value,3)==-0.931)
    return True
U1, Sigma, V1 = randomized_svd(adjacency_matrix, n_components=2,n_iter=5, random_state=24)
# Please don't change random state
# Here we are considering n_componets = 2 for our convinence
alpha=0.01
value=derivative_db(312,98,4,U1,V1,mu,alpha)
grader_db(value)
```

Out[69]:

```
True
```

Compute dL/dc_j

In [79]:

```
def derivative_dc(user_id,item_id,rating,U,V,mu, alpha):
    '''In this function, we will compute dL/dc_j'''
    bi = b_i[user_id]
    bi_2 = bi ** 2
    u = U[user_id]
    v = V[:,item_id]
    cj = c_j[item_id]
    cj_2 = cj ** 2
    y_ij = rating

    value = (2 * alpha * cj * (np.dot(v,v) + np.dot(u,u) + bi_2 + cj_2)) - (2 * (y_ij - mu - bi - cj - (np.
    return value
```

Grader function - 5

In [80]:

```
def grader_dc(value):
    assert(np.round(value,3)==-2.929)
    return True
U1, Sigma, V1 = randomized_svd(adjacency_matrix, n_components=2,n_iter=5, random_state=24)
# Please don't change random state
# Here we are considering n_componets = 2 for our convinence
r=0.01
value=derivative_dc(58,504,5,U1,V1,mu, alpha)
grader_dc(value)
```

Out[80]:

```
True
```

Compute MSE (mean squared error) for predicted ratings

for each epoch, print the MSE value

```
for each epoch:

    for each pair of (user, movie):

        b_i =  b_i - learning_rate * dL/db_i

        c_j =  c_j - learning_rate * dL/dc_j

predict the ratings with formula
```

$\hat{y}_{ij} = \mu + b_i + c_j + \text{dot_product}(u_i , v_j) $

In [87]:

```
from tqdm import tqdm
```

In [100]:

```
def apply_sgd(r, alpha):
    epoch = 50
    epoch_mse = dict()
    for i in tqdm(range(epoch)):
        for index, row in data.iterrows():
            user_id = row['user_id']
            item_id = row['item_id']
            rating = row['rating']

            b_i[user_id] = b_i[user_id] - (r * derivative_db(user_id, item_id, rating, U, VT, mu, alpha))
            c_j[item_id] = c_j[item_id] - (r * derivative_dc(user_id, item_id, rating, U, VT, mu, alpha))

        y_pred = []
        for index, row in data.iterrows():
            user_id = row['user_id']
            item_id = row['item_id']
            u_i = U[user_id]
            v_j = VT[:,item_id]
            y_pred_ij = mu + b_i[user_id] + c_j[item_id] + np.dot(u_i,v_j)
            y_pred.append(y_pred_ij)

        y = data['rating'].values
        y_pred = np.array(y_pred)
        mse = np.mean(np.square(y - y_pred))
        epoch_mse[i] = mse
    return epoch_mse
```

In [101]:

```
mse_result = apply_sgd(0.0001, 0.001)
```

```
100%|████████████████████████████████████████| 50/50 [32:44<00:
00, 39.30s/it]
```

Plot epoch number vs MSE

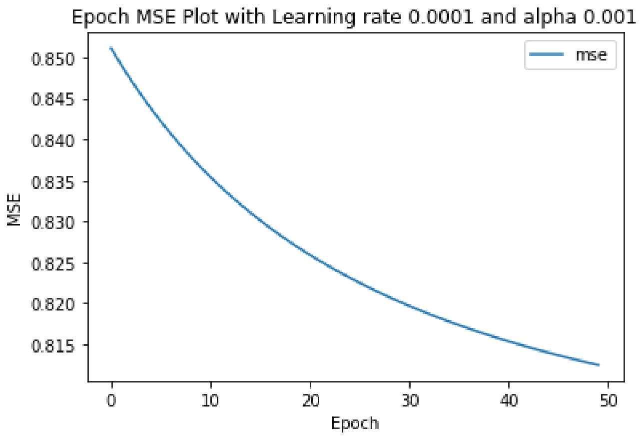
- epoch number on X-axis
- MSE on Y-axis

In [94]:

```
import matplotlib.pyplot as plt
```

In [102]:

```
plt.plot(list(mse_result.keys()), list(mse_result.values()), label = 'mse')
plt.xlabel('Epoch')
plt.ylabel('MSE')
plt.legend()
plt.title("Epoch MSE Plot with Learning rate {} and alpha {}".format(0.0001, 0.001))
plt.show()
```




Task 2

In [138]:

```
user_data=pd.read_csv('user_info.csv')
print(user_data.shape)
user_data.head()
```

```
(943, 4)
```

Out[138]:

| | user_id | age | is_male | orig_user_id |
|---|---|---|---|---|
| **0** | 0 | 24 | 1 | 1 |
| **1** | 1 | 53 | 0 | 2 |
| **2** | 2 | 23 | 1 | 3 |
| **3** | 3 | 24 | 1 | 4 |
| **4** | 4 | 33 | 0 | 5 |

In [139]:

```
user_data.drop(['user_id', 'age'], axis=1, inplace=True)
user_data.head()
```

Out[139]:

| | is_male | orig_user_id |
|---|---|---|
| **0** | 1 | 1 |
| **1** | 0 | 2 |
| **2** | 1 | 3 |
| **3** | 1 | 4 |
| **4** | 0 | 5 |

In [140]:

```
user_data.rename(columns = {'orig_user_id': 'user_id'}, inplace=True)
user_data.head()
```

Out[140]:

| | is_male | user_id |
|---|---|---|
| **0** | 1 | 1 |
| **1** | 0 | 2 |
| **2** | 1 | 3 |
| **3** | 1 | 4 |
| **4** | 0 | 5 |

In [148]:

```
user_gender = pd.merge(data, user_data, on= 'user_id', how= 'left')
user_gender.head()
```

Out[148]:

| | user_id | item_id | rating | is_male |
|---|---|---|---|---|
| **0** | 772 | 36 | 3 | 1.0 |
| **1** | 471 | 228 | 5 | 1.0 |
| **2** | 641 | 401 | 4 | 1.0 |
| **3** | 312 | 98 | 4 | 1.0 |
| **4** | 58 | 504 | 5 | 1.0 |

In [158]:

```
user_gender.isnull().any()
```

Out[158]:

```
user_id     False
item_id     False
rating      False
is_male      True
dtype: bool
```

In [159]:

```
user_gender.dropna(inplace=True)
```

In [160]:

```
user_gender.isnull().any()
```

Out[160]:

```
user_id    False
item_id    False
rating     False
is_male    False
dtype: bool
```

In [165]:

```
user_gender.astype('int64').head()
```

Out[165]:

| | user_id | item_id | rating | is_male |
|---|---|---|---|---|
| 0 | 772 | 36 | 3 | 1 |
| 1 | 471 | 228 | 5 | 1 |
| 2 | 641 | 401 | 4 | 1 |
| 3 | 312 | 98 | 4 | 1 |
| 4 | 58 | 504 | 5 | 1 |

In [178]:

```
user_gender['is_male'].value_counts()
```

Out[178]:

```
1.0    62985
0.0    26764
Name: is_male, dtype: int64
```

In [185]:

```
user_matrix = np.zeros((data.shape[0], 500))
gender = np.zeros(data.shape[0], dtype='int64')
print(user_matrix.shape)
print(gender.shape)
```

```
(89992, 500)
(89992,)
```

In [186]:

```
for index, row in user_gender.iterrows():
    user_id = int(row['user_id'])
    user_matrix[user_id] = U[user_id]
    gender[user_id] = int(row['is_male'])

print(user_matrix[772,:10])
print(gender[772])
```

```
[ 0.03525135  0.01588283  0.01476179 -0.06828845  0.05270653  0.01506547
  0.08118817  0.00115754 -0.02210448  0.02467522]
1
```

**Apply model**

In [201]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

In [202]:

```
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)
    print("Number of misclassified points ",(len(test_y)-np.trace(C))/len(test_y)*100)
    A =(((C.T)/(C.sum(axis=1))).T)
    B =(C/C.sum(axis=0))
    labels = [1,2]
    cmap=sns.light_palette("green")

    print("-"*50, "Confusion matrix", "-"*50)
    plt.figure(figsize=(10,5))
    sns.heatmap(C, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()

    print("-"*50, "Precision matrix", "-"*50)
    plt.figure(figsize=(10,5))
    sns.heatmap(B, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
```

```
    plt.show()
    print("Sum of columns in precision matrix",B.sum(axis=0))


    print("-"*50, "Recall matrix"    , "-"*50)
    plt.figure(figsize=(10,5))
    sns.heatmap(A, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, yticklabels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.show()
    print("Sum of rows in precision matrix",A.sum(axis=1))
```

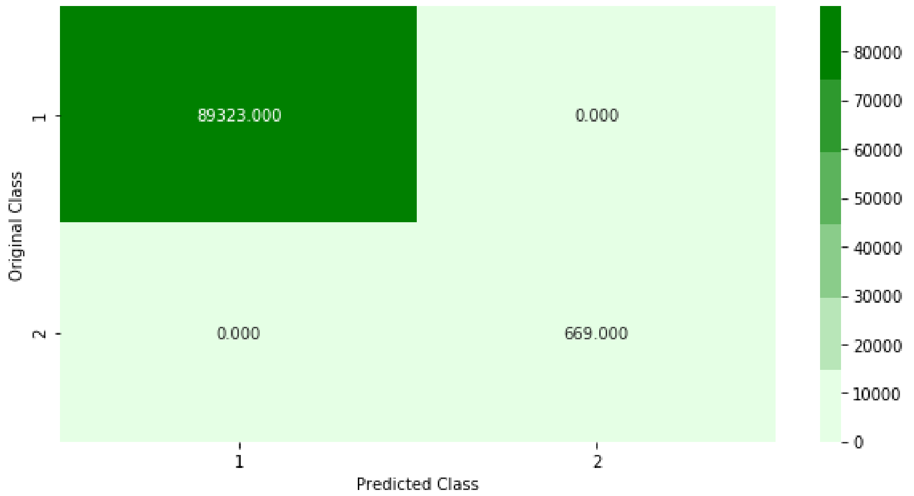
In [197]:

```
clf = RandomForestClassifier(n_estimators=100, class_weight='balanced', random_state=0)
clf.fit(user_matrix, gender)
gender_pred = clf.predict(user_matrix)
```

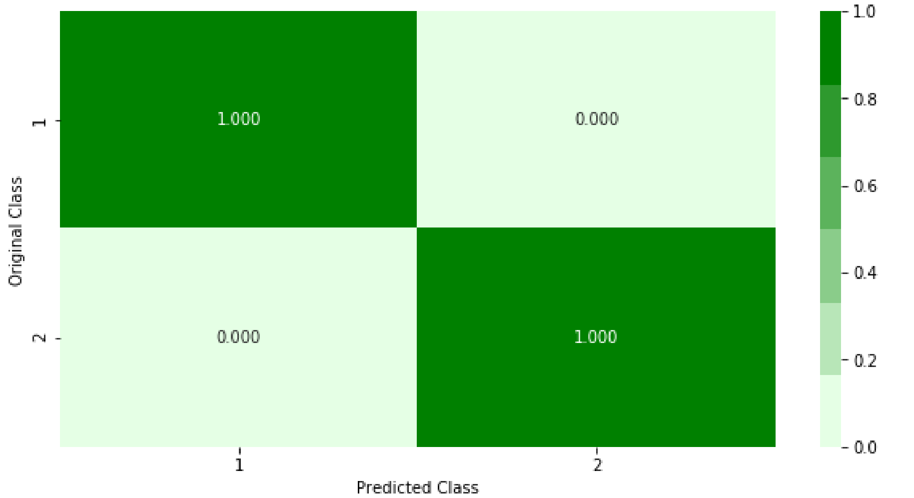
In [203]:

```
plot_confusion_matrix(gender, gender_pred)
```

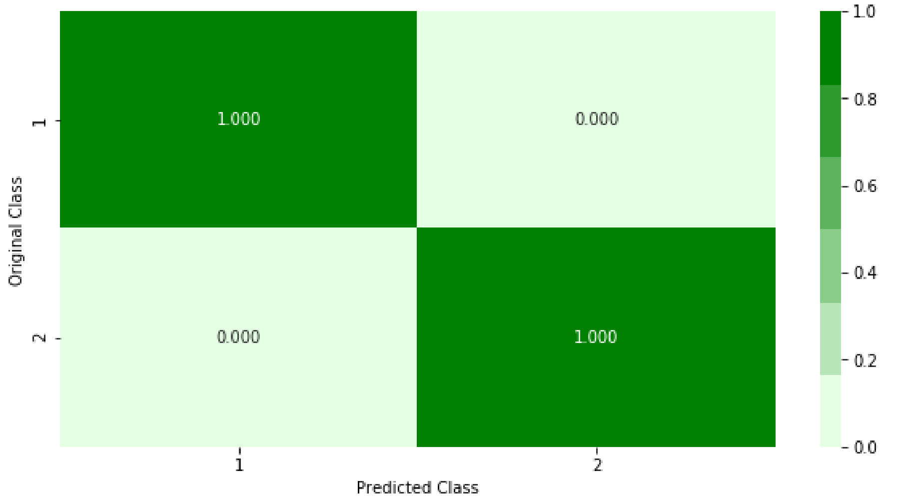
```
Number of misclassified points  0.0
-------------------------------------------------- Confusion matrix -----------------------------------
---------
```




```
-------------------------------------------------- Precision matrix -----------------------------------
---------
```




```
Sum of columns in precision matrix [1. 1.]
-------------------------------------------------- Recall matrix ---------------------------------------
------
```

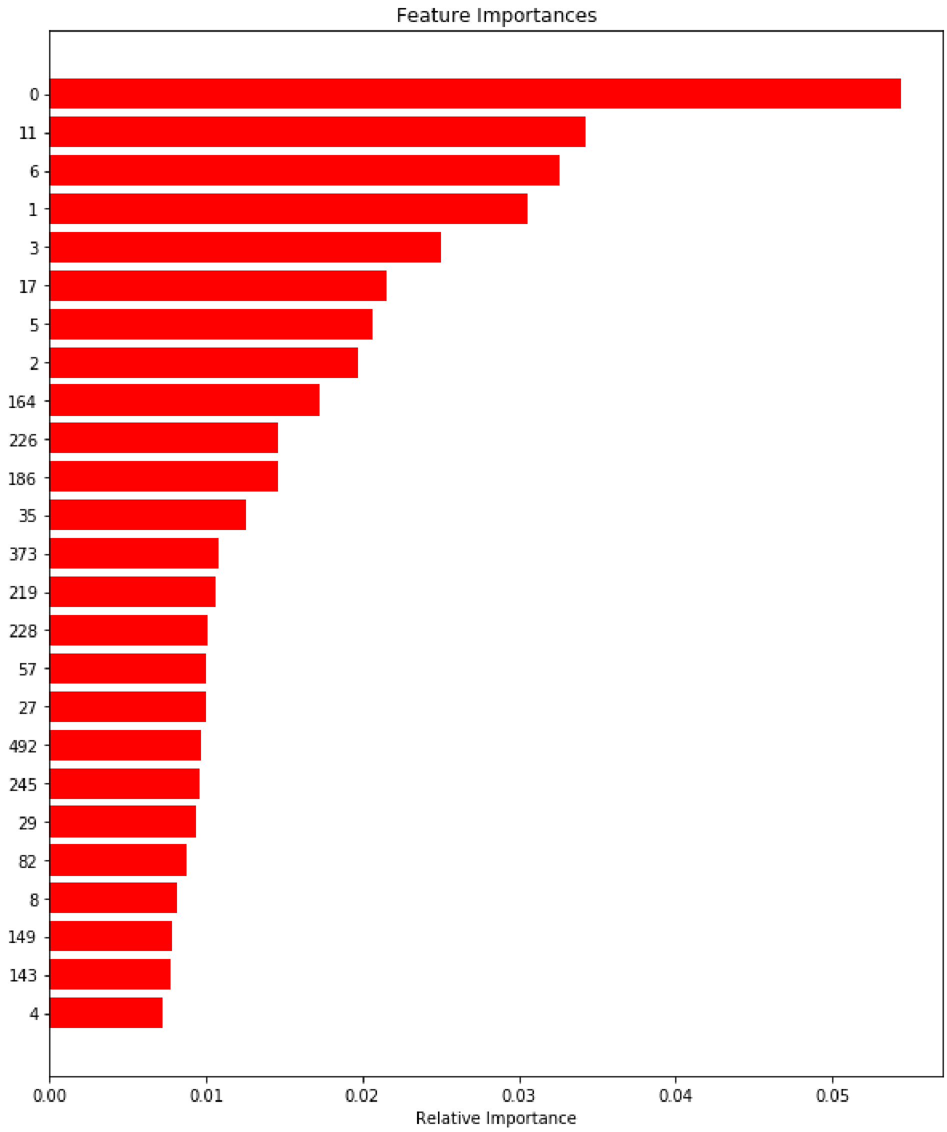



```
Sum of rows in precision matrix [1. 1.]
```

In [206]:

```
importances = clf.feature_importances_
indices = (np.argsort(importances))[-25:]
plt.figure(figsize=(10,12))
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='r', align='center')
plt.yticks(range(len(indices)), indices)
```

```
plt.xlabel('Relative Importance')
plt.show()
```




We can see the user features are perfectly classifying the gender. Here all training data are used to train and test. With training set, the model is showing very good result.

We can also see, as here only user features are used and some of them are highly important features to determine gender.

So definitly these features decribe the user.