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_

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

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

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

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

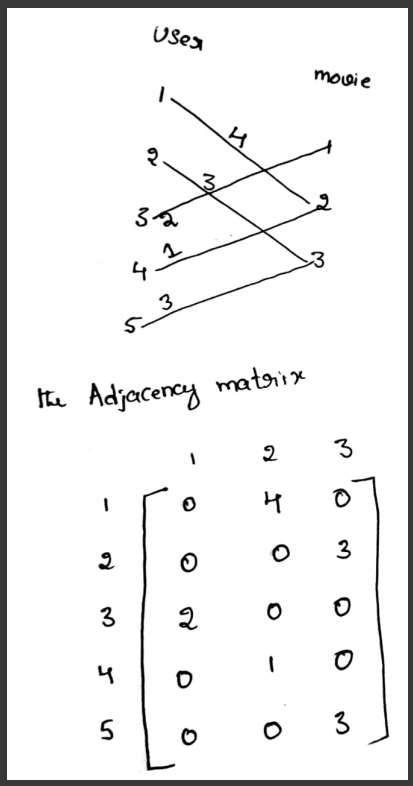
- μ : scalar mean rating
- b_i : scalar bias term for user i

- 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 graph and the weight of each edge is the rating given by user to the movie



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

2. 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.

3. Compute μ , μ represents the mean of all the rating given in the dataset.(write your code in def m_u())
4. 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())
5. 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())
6. Compute dL/db_i (Write you code in def derivative_db())
7. Compute dL/dc_j(write your code in def derivative_dc()
8. 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
```

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

9. you can choose any learning rate and regularization term in the range 10^{-3} to 10^{2}
10. **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

```
import pandas as pd
data=pd.read_csv('/content/drive/MyDrive/AI-ML-Assignments/Recommendation System using Tru
data.head()
```

	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

```
data.shape
```

```
(89992, 3)
```

Create your adjacency matrix

```
import numpy as np
np.array(data['item_id'])
```

```
array([ 36, 228, 401, ..., 505, 472, 204])
```

```
import numpy as np
```

```
from scipy.sparse import csr_matrix
ratings = np.array(data['rating'])
users = np.array(data['user_id'])
items = np.array(data['item_id'])
adjacency_matrix = csr_matrix((ratings,(users,items)))
```

```
adjacency_matrix.shape
```

```
(943, 1681)
```

Grader function - 1

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

```
True
```

The unique items in the given csv file are 1662 only . But the id's vary from 0-1681 but they are not continuous and hence you'll get matrix of size 943x1681.

SVD decompostion

Sample code for SVD decompostion

```
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

```
# Please use adjacency_matrix as matrix for SVD decompostion
# You can choose n_components as your choice
U, Sigma, VT = randomized_svd(adjacency_matrix, n_components=20,n_iter=5, random_state=167
print(U.shape)
print(Sigma.shape)
print(VT.shape)
```

```
(943, 20)
(20,)
(20, 1681)
```

```
print(U.shape)
```

```
(943, 20)
```

Compute mean of ratings

```
import statistics

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.DataFr
    return statistics.mean(ratings)
```

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

```
3.529480398257623
```

Grader function -2

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

```
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)

```
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
    return np.zeros(dim)
```

```
#dim= give the number of dimensions for b_i (Here b_i corresponds to users)
dim = 943
```

```
b_i=initialize(dim)
```

```
#dim=  give the number of dimensions for c_j (Here c_j corresponds to movies)
dim = 1681
c_j=initialize(dim)
```

Grader function -3

```
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)
```

```
True
```

Compute dL/db_i

```
def derivative_db(user_id,item_id,rating,U,VT,mu,alpha):
    '''In this function, we will compute dL/db_i'''
    dL_db_i = 2 * (b_i[user_id]) * alpha - 2*(rating - mu - b_i[user_id] - c_j[item_id] -
    return dL_db_i
```

Grader function -4

```
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)
```

```
True
```

```
```

Compute dL/dc_j

```
def derivative_dc(user_id,item_id,rating,U,V,mu, alpha):
    '''In this function, we will compute dL/dc_j'''
    dL_dc_j = 2 * (c_j[item_id]) * alpha - 2*(rating - mu - b_i[user_id] - c_j[item_id] -
    return dL_dc_j
```

Grader function - 5

```
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)
```

```
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
```

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

```
u0 = data['user_id'][0]
m0 = data['item_id'][0]
r0 = data['rating'][0]
```

```
b_i[u0] = b_i[u0] - 0.01 * derivative_db(u0,m0,r0,U,VT,mu,alpha)
c_j[m0] = c_j[m0] - 0.01 * derivative_dc(u0,m0,r0,U,VT,mu,alpha)
```

```
U[u0,:]
```

```
array([ 0.03525135,  0.01588283,  0.01476178, -0.06828861,  0.05270592,
        0.01506997,  0.08118772,  0.00107447, -0.02226247,  0.02542742,
       -0.01496991, -0.00639124, -0.02387486, -0.02676716,  0.03310974,
        0.03283066,  0.00834765,  0.07460381,  0.02551229, -0.00065034])
```

```
VT[:,m0]
```

```
array([ 1.74536808e-03,  1.67829433e-03,  7.78886589e-04, -5.81425788e-03,
        7.49132421e-03,  2.46167669e-03,  3.54860145e-03,  4.12387673e-03,
       -9.87028645e-05, -2.04819960e-03,  8.04031559e-04,  2.61485557e-03,
        1.53869224e-03,  4.18519466e-03,  5.88949206e-03,  4.65622745e-03,
        3.62784921e-03, -6.48066539e-04,  4.91381293e-03, -2.46384613e-04])
```

```
y_pred = mu + b_i[u0] + c_j[m0] + np.dot(U[u0,:],VT[:,m0])
```

```
r0
```

```
3
```

```
print(y_pred)
print(r0-y_pred)
```

```
3.5099047172142583
-0.5099047172142583
```

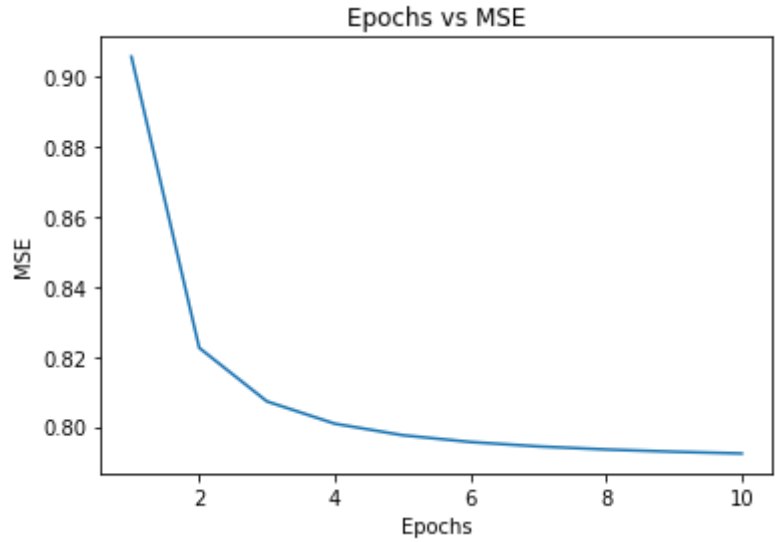
```
from tqdm import tqdm
from sklearn.metrics import mean_squared_error
def compute_mean_sq_error(data,U,V,mu,alpha):
    y_actual = data['rating']
    mean_sq_error, y_predict = [],[]
    for epoch in range(10):
      for data_point in tqdm(np.array(data)):
            user_id = data_point[0]
            item_id = data_point[1]
            rating = data_point[2]
            b_i[user_id] = b_i[user_id] - alpha * derivative_db(user_id,item_id,rating,U,V
            c_j[item_id] = c_j[item_id] - alpha * derivative_dc(user_id,item_id,rating,U,V
            y_predict.append(mu + b_i[data_point[0]] + c_j[data_point[1]] + np.dot(U[data_
      mean_sq_error.append(mean_squared_error(y_actual,y_predict))
      y_predict *= 0
    return mean_sq_error
```

```
MSE = compute_mean_sq_error(data,U,VT,mu,alpha)
print(MSE)
```

```
100%|██████████| 89992/89992 [13:50<00:00, 108.41it/s]
100%|██████████| 89992/89992 [13:17<00:00, 112.85it/s]
100%|██████████| 89992/89992 [13:50<00:00, 108.32it/s]
100%|██████████| 89992/89992 [13:42<00:00, 109.45it/s]
100%|██████████| 89992/89992 [13:44<00:00, 109.20it/s]
100%|██████████| 89992/89992 [13:49<00:00, 108.48it/s]
100%|██████████| 89992/89992 [14:16<00:00, 105.01it/s]
100%|██████████| 89992/89992 [14:23<00:00, 104.22it/s]
100%|██████████| 89992/89992 [14:07<00:00, 106.20it/s]
100%|██████████| 89992/89992 [13:51<00:00, 108.20it/s][0.9055630191591441, 0.8225265
```

```
import matplotlib.pyplot as plt

epochs = np.arange(1,11,1)
plt.plot(epochs,MSE)
plt.xlabel('Epochs')
plt.ylabel('MSE')
plt.title('Epochs vs MSE')
```

```
Text(0.5, 1.0, 'Epochs vs MSE')
```



Task 2

- For this task you have to consider the user_matrix U and the user_info.csv file.
- You have to consider is_male columns as output features and rest as input features. Now you have to fit a model by posing this problem as binary classification task.
- You can apply any model like Logistic regression or Decision tree and check the performance of the model.
- Do plot confusion matrix after fitting your model and write your observations how your model is performing in this task.
- Optional work- You can try scaling your U matrix.Scaling means changing the values of n_componenets while performing svd and then check your results.

```
import pandas as pd
user=pd.read_csv('/content/drive/MyDrive/AI-ML-Assignments/Recommendation System using Tru
user.head()
```

	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

```
U
```

```
array([[ 0.0662257 ,  0.00788857, -0.01253133, ..., -0.02083788,
        -0.06543583, -0.09958412],
       [ 0.01364432, -0.04889503,  0.05655366, ...,  0.01483133,
         0.00731117,  0.00162532],
       [ 0.00543826, -0.02512779,  0.02002774, ..., -0.01051827,
        -0.0253918 ,  0.00665187],
       ...,
       [ 0.00738924, -0.02597375,  0.00634334, ..., -0.0342234 ,
         0.03405594, -0.0108061 ],
       [ 0.02499924,  0.00447792,  0.02605655, ..., -0.02317566,
         0.04409229,  0.00455369],
       [ 0.04337341, -0.00281488, -0.06077791, ...,  0.01573796,
         0.0180218 ,  0.02950242]])
```

```
for i in range(1,21):
  user[str('f_')+str(i)] = 0
```

```
user.head()
```

	user_id	age	is_male	orig_user_id	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
0	0	24	1	1	0	0	0	0	0	0	0	0	0
1	1	53	0	2	0	0	0	0	0	0	0	0	0
2	2	23	1	3	0	0	0	0	0	0	0	0	0
3	3	24	1	4	0	0	0	0	0	0	0	0	0
4	4	33	0	5	0	0	0	0	0	0	0	0	0

```
user.loc[0,'f_1':'f_20']
```

```
f_1     0
f_2     0
f_3     0
f_4     0
f_5     0
f_6     0
f_7     0
f_8     0
f_9     0
f_10    0
f_11    0
f_12    0
```

```
f_13    0
f_14    0
f_15    0
f_16    0
f_17    0
f_18    0
f_19    0
f_20    0
Name: 0, dtype: int64
```

```
U[0]
```

```
array([ 0.0662257 ,  0.00788857, -0.01253133, -0.08615456,  0.02486547,
        0.0066645 ,  0.07997384, -0.02743249,  0.0686735 ,  0.0204221 ,
       -0.02142243,  0.02551735, -0.0366062 ,  0.03772546,  0.01355219,
        0.00535297,  0.09296851, -0.02083788, -0.06543583, -0.09958412])
```

```
for i in range(0,len(user)):
  user.loc[i,'f_1':'f_20'] = U[i]
```

```
user.drop(['user_id','orig_user_id'],axis=1,inplace=True)
```

```
user
```

	age	is_male	f_1	f_2	f_3	f_4	f_5	f_6	
0	24	1	0.066226	0.007889	-0.012531	-0.086155	0.024865	0.006664	0.07
1	53	0	0.013644	-0.048895	0.056554	0.015809	-0.012034	0.017736	0.01
2	23	1	0.005438	-0.025128	0.020028	0.032831	0.035080	0.001918	0.00
3	24	1	0.005704	-0.018211	0.010898	0.021867	0.013919	-0.014195	0.01
4	33	0	0.034122	0.009005	-0.044054	-0.016050	0.004328	-0.021504	0.09
...	...	...	...	...	...	...	...	...	
938	26	0	0.010350	-0.038006	0.006501	-0.013992	-0.051220	-0.001720	-0.03
939	32	1	0.031624	-0.007730	0.032983	0.013861	0.023618	-0.008445	0.05
940	20	1	0.007389	-0.025974	0.006343	-0.017067	-0.007398	-0.020793	0.01
941	48	0	0.024999	0.004478	0.026057	0.077342	-0.000768	-0.038294	-0.01
942	22	1	0.043373	-0.002815	-0.060778	-0.031584	0.039834	0.006383	-0.04

943 rows × 22 columns

```
x = user.drop('is_male',axis=1)
y = user['is_male']
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.33)
```

```
print(x_train.shape)
print(y_train.shape)
```

```
(631, 21)
(631,)
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(C=1.0, random_state=1679)
model.fit(x_train,y_train)
```

```
LogisticRegression(random_state=1679)
```

```
y_pred = model.predict(x_test)
```

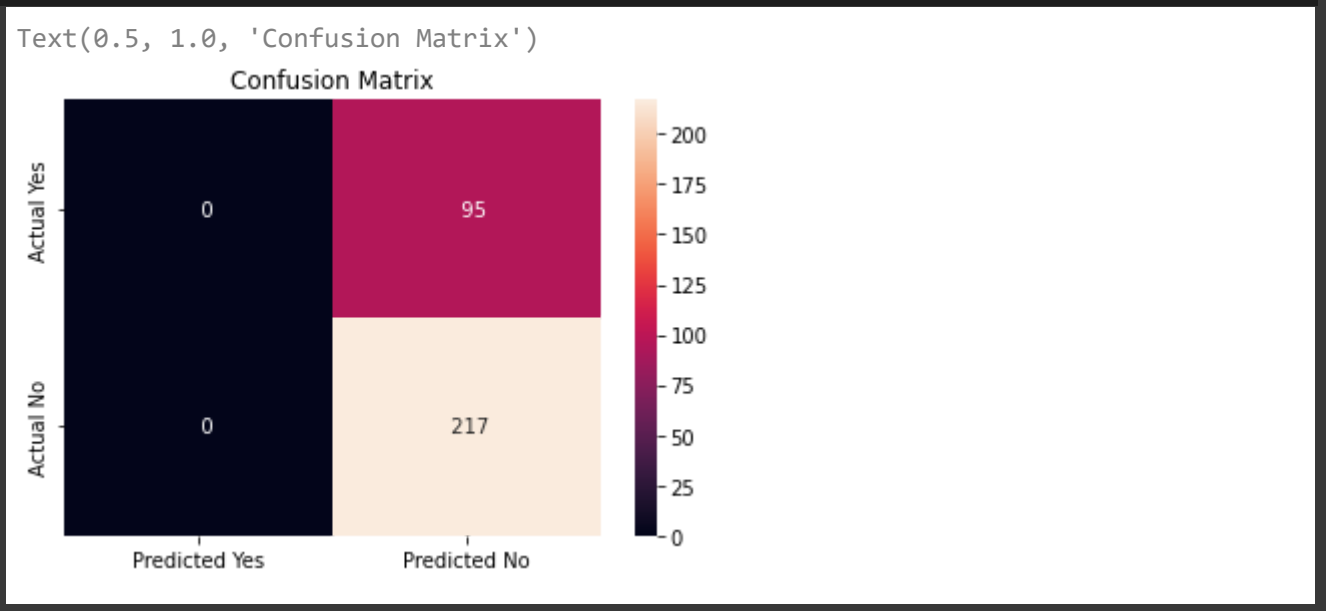
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,y_pred)
```

```
cm
```

```
array([[  0,  95],
       [  0, 217]])
```

```
import seaborn as sns
sns.heatmap(cm,annot=True,fmt='4d', xticklabels=['Predicted Yes','Predicted No'],yticklabe
plt.title('Confusion Matrix')
```

```
Text(0.5, 1.0, 'Confusion Matrix')
```

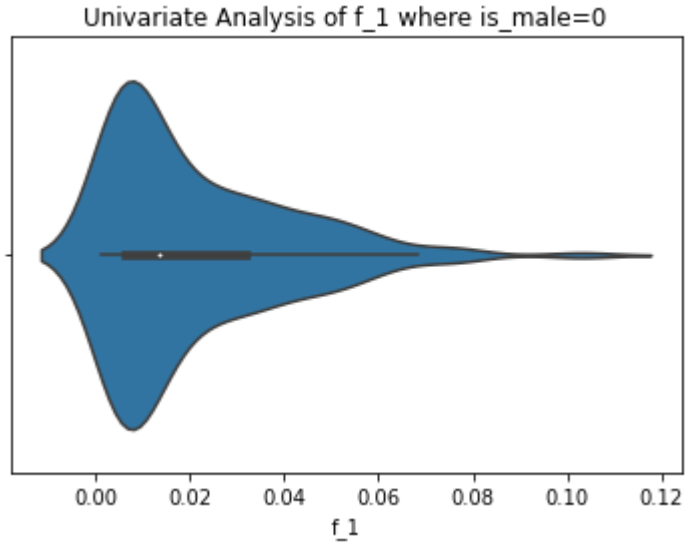



```
f_1_is_not_male = user[user['is_male']==0]['f_1']
```

```
f_1_is_male = user[user['is_male']!=0]['f_1']
```

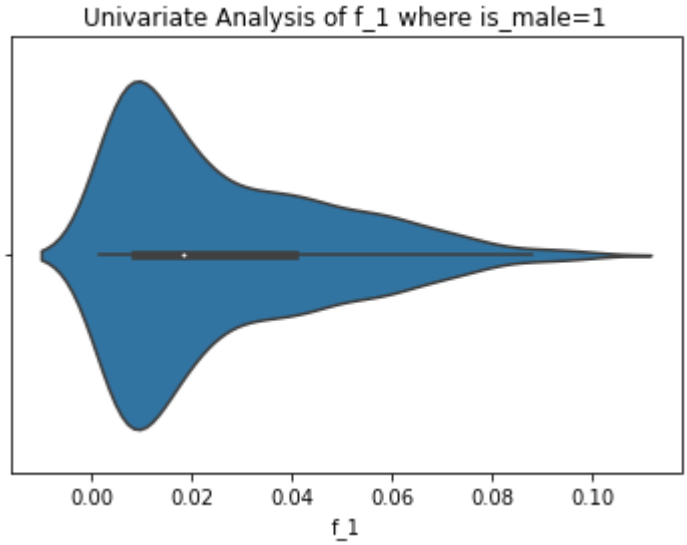
```
sns.violinplot(x = f_1_is_not_male)
plt.title('Univariate Analysis of f_1 where is_male=0')
```

```
Text(0.5, 1.0, 'Univariate Analysis of f_1 where is_male=0')
```




```
sns.violinplot(x = f_1_is_male)
plt.title('Univariate Analysis of f_1 where is_male=1')
```

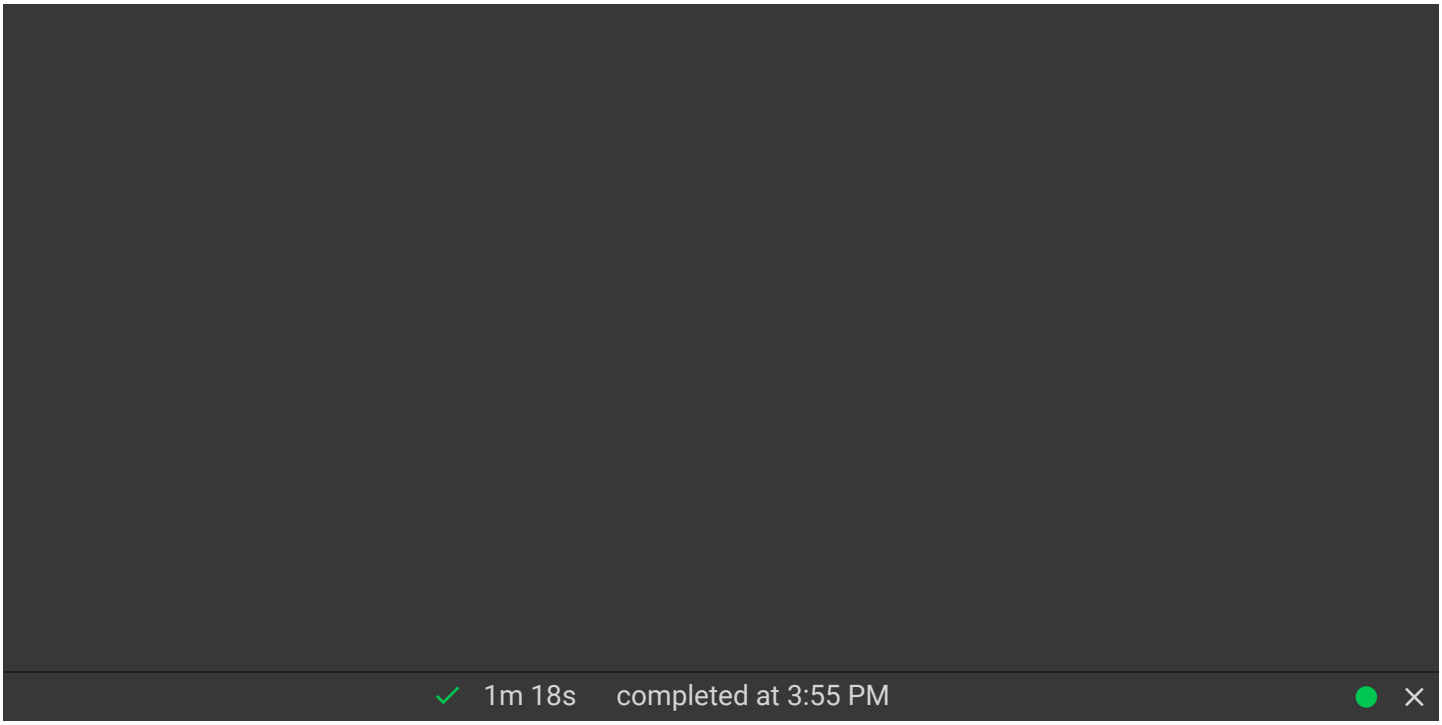
```
Text(0.5, 1.0, 'Univariate Analysis of f_1 where is_male=1')
```



For example, here we took feature f_1 w.r.to target, is_male=0, is_male!=0

The above plot is the violin plot of f_1 of these two categories.

There is no major difference between the distribution of data between these two categories. Henceforth Logistic regression cant abe to study the inferences between the features, hence predicting everything as negative


1m 18s completed at 3:55 PM