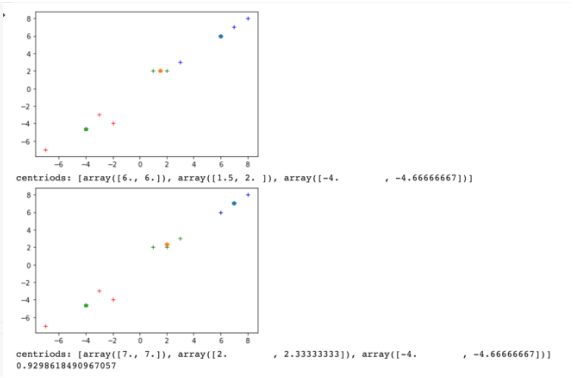
T1. If the starting points are (3,3), (2,2), and (-3,-3). Describe each assign and update step. What are the points assigned? What are the updated centroids? You may do this calculation by hand or write a program to do it.

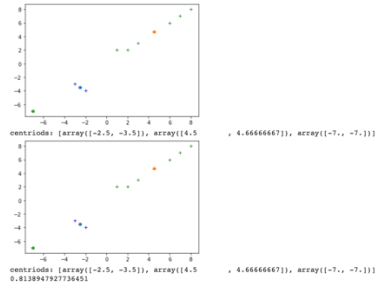
```
# Assign centroids
def k_mean(c0, c1, c2):
  centroids = [c0, c1, c2]
  # list for keeping k cluster for each data
  k = [0]*10
  # finding the cluster for each data
  old_centriods = list(np.zeros((1, 2)))*4
  while(1):
    member_in_cluster = {0:[], 1:[], 2:[]}
    for index, point in enumerate(data):
      dis_list = []
      for centroid in centroids:
        dis = np.sqrt(np.sum((point - centroid)**2))
        dis_list.append(dis)
      cluster = np.argmin(dis_list)
      k[index] = cluster
      member_in_cluster[cluster].append(point)
    # updating the centroids
    for cluster, members in member_in_cluster.items():
      members = np.array(members)
      plt.plot(members[:, 0], members[:, 1], ['b+', 'g+', 'r+'][cluster])
      centroids[cluster] = np.average(members, axis=0)
    for c in centroids:
      plt.plot(c[0], c[1], 'p-')
    plt.show()
    print(f'centriods: {centroids}')
    if old_centriods == centroids:
      return old_centriods, member_in_cluster
    old_centriods = centroids
```



From the pictures, we can see that there are two steps in order to fit the model. Points are grouped together likes [(-7, -7), (-2, -4), (-3, -3)] [(1, 2), (2, 2), (3, 3)] [(6, 6), (7, 7), (8, 8)] as we can see in the second plot. The centroids will be shown in the image as well.

T2. If the starting points are (-3,-3), (2,2), and (-7,-7), what happens?
The cluster is different from the first trial as you can see in the following image.

```
c0 = (-3, -3)
c1 = (2, 2)
c2 = (-7, -7)
centroids, members = k_mean(c0, c1, c2)
print(explained_var(centroids, members))
```



T3. Between the two starting set of points in the previous two questions, which one do you think is better? How would you measure the ‘goodness’ quality of a set of starting points?

In my opinion, the first one, by using the explainable variance, the first one accounts for 0.929 while the second one’s is 0.814. With this metric, I believe that we should consider using the first case as the starting point for centroids.

```
def between_cluster_var(centroids, member_in_cluster):
  all_data_centroid = np.average(data, axis=0)
  var = 0
  N = 10
  for cluster_i, members in member_in_cluster.items():
    members = np.array(members)
    cluster_centroid = np.average(members, axis=0)
    var += (len(members)*np.sum((cluster_centroid - all_data_centroid)**2))/(N-1)
  # print(f'between_cluster_var: {var}')
  return var

def get_var():
  all_data_centroid = np.average(data, axis=0)
  N = 10
  var = np.sum((data - all_data_centroid)**2)/(N-1)
  return var

def explained_var(centroids, member_in_cluster):
  return between_cluster_var(centroids, member_in_cluster) / get_var()
```

T4. What is the median age of the training set? You can easily modify the age in the dataframe by train["Age"] = train["Age"].fillna(train["Age"].median())
28

```
[151] def impute(df):
        train_med_age = df["Age"].median()
        df["Age"] = df["Age"].fillna(train_med_age)
        print(f'train median age: {train_med_age}')

        df.loc[df["Embarked"] == "S", "Embarked"] = 0
        df.loc[df["Embarked"] == "C", "Embarked"] = 1
        df.loc[df["Embarked"] == "Q", "Embarked"] = 2
        train_mod_embark = list(df["Embarked"].mode())[0]
        print(f'mode embarked: {train_mod_embark}')
        df["Embarked"] = df["Embarked"].fillna(train_mod_embark)

        df.loc[df["Sex"] == "male", "Sex"] = 0
        df.loc[df["Sex"] == "female", "Sex"] = 1
        train_mod_sex = list(df["Sex"].mode())[0]
        df["Sex"] = df["Sex"].fillna(train_mod_sex)
        return df
```

```
[152] train = impute(train)
      train

      train median age: 28.0
      mode embarked: 0
```

Note that you need to modify the code above a bit to fill with mode() because mode() returns a series rather than a single value.

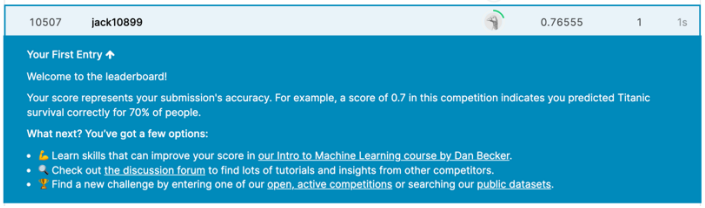
T5. Some fields like ‘Embarked’ are categorical. They need to be converted to numbers first. We will represent S with 0, C with 1, and Q with 2. What is the mode of Embarked? Fill the missing values with the mode. You can set the value of Embarked easily with the following command. train.loc[train["Embarked"] == "S", "Embarked"] = 0 Do the same for Sex.
0 from the above image.

T6. Write a logistic regression classifier using gradient descent as learned in class. Use PClass, Sex, Age, and Embarked as input features.

```
def h(value):
  return 1/(1+np.exp(-value))
def logistic_regression(param_num, learning_rate, epoch, data):
  theta = np.array([0.5]*param_num)
  learning_rate = learning_rate
  for e in range(epoch):
    for i in range(len(data)):
      predict = h(data[i].dot(theta))
      gradient = learning_rate*(y[i]-predict)*data[i]
      theta += gradient
    error = np.sqrt(np.sum((y - h(data.dot(theta)))**2))
    if e % 50 == 0:
      print(error)
  return theta
```

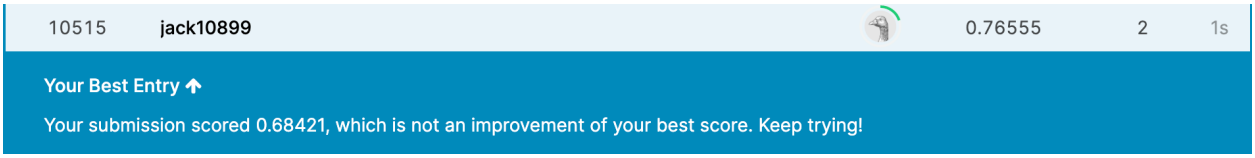
```
] theta = logistic_regression(4, 0.05, 1000, data)

 18.484918325542413
 14.673949725348512
 14.704992138214271
 14.644423266759844
 14.668876477677752
 14.654855396817618
 14.59301284261833
 14.338616912114345
 14.662118788670375
 14.671981998322583
 14.421384001347912
 13.18258375849629
 14.69419182222592
 14.369876605447338
 14.656182361583236
 13.964615996265897
 14.701818029422498
 14.64829692705079
 14.357547662063249
 14.699229207913515
```

T7. Submit a screenshot of your submission (with the scores). Upload your code to courseville.



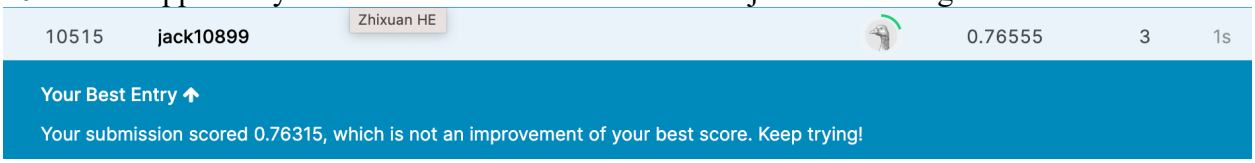
T8. Try adding some higher order features to your training (x 2 1 , x1x2,...). Does this model has better accuracy on the training set? How does it perform on the test set?

```
data2 = train[["Pclass","Sex","Age","Embarked"]]
data2['Age_squared'] = data2['Age']**2
data2['Sex_times_Age'] = data2['Sex'] * data2['Age']
data2_np = np.array(data2[["Pclass","Sex","Age","Embarked", 'Age_squared', 'Sex_times_Age']].values, dtype = float)
y = np.array(train['Survived'])
theta = logistic_regression(6, 0.05, 1000, data2_np)
```



I think it worse than not doing anything.

T9. What happens if you reduce the amount of features to just Sex and Age?



Still lost to the first attempt, but after dropping two features, the score is very close to the first one. This is interesting because the other two features might not be that necessary when predict if the one is survived from Titanic.