Foundations of Data Science Fall 2022 - Homework 2 (30 points)

Student Name: Yamini Lakshmi Narasimhan

Student Net Id: yl9822

Part 1: Preparing a Training Set and Training a Decision Tree (10 Points)

This is a hands-on task where we build a predictive model using Decision Trees discussed in class. For this part, we will be using the data in `cell2cell_data.csv` (you can find this on NYU Brightspace).

These historical data consist of 39,859 customers: 19,901 customers that churned (i.e., left the company) and 19,958 that did not churn (see the `"churndep"` variable). Here are the data set's 11 possible predictor variables for churning behavior:

```
Pos.  Var. Name  Var. Description
----- ---------- ----------------------------------------------
----------------
1     revenue    Mean monthly revenue in dollars
2     outcalls   Mean number of outbound voice calls
3     incalls    Mean number of inbound voice calls
4     months     Months in Service
5     eqpdays    Number of days the customer has had his/her
current equipment
6     webcap     Handset is web capable
7     marryyes   Married (1=Yes; 0=No)
8     travel     Has traveled to non-US country (1=Yes; 0=No)
9     pcown      Owns a personal computer (1=Yes; 0=No)
10    creditcd   Possesses a credit card (1=Yes; 0=No)
11    retcalls   Number of calls previously made to retention
team
```

The 12th column, the dependent variable `"churndep"`, equals 1 if the customer churned, and 0 otherwise.

In [1]:
```
import warnings
from pprint import pprint
warnings.filterwarnings('ignore')
warnings.filterwarnings(action='once')
```

1. Load the data and prepare it for modeling. Note that the features are already processed

for you, so the only thing needed here is split the data into training and testing. Use pandas to create two data frames: train_df and test_df, where train_df has 80% of the data chosen uniformly at random without replacement (test_df should have the other 20%). Also, make sure to write your own code to do the splits. You may use any random() function numpy but do not use the data splitting functions from Sklearn.

(2 Points)

In [2]:
```
# Place your code here
import pandas as pd
import numpy as np

df = pd.read_csv("./cell2cell_data.csv")

print("Total data",len(df))
percent = np.random.rand(len(df)) <= 0.8
train_df = df[percent]
test_df = df[~percent]

print("Train df",len(train_df))
print("Test df",len(test_df))

train_X = train_df.loc[:, train_df.columns != "churndep"]
train_Y = train_df["churndep"]


test_X = test_df.loc[:, test_df.columns != "churndep"]
test_Y = test_df["churndep"]
```

```
Total data 39833
Train df 31951
Test df 7882
```

In [3]:
```
print(train_df["churndep"].value_counts())
print(test_df["churndep"].value_counts())
```

```
0    15977
1    15974
Name: churndep, dtype: int64
0    3973
1    3909
Name: churndep, dtype: int64
```

In [4]:
```
df.head()
```

Out[4]:

	revenue	outcalls	incalls	months	eqpdays	webcap	marryyes	travel	pcown	creditcd	re
0	48.82	10.00	3.0	26	780	0	0	0	0	1	
1	83.53	20.00	1.0	31	745	1	0	0	0	0	
2	29.99	0.00	0.0	52	1441	0	0	0	1	1	
3	51.42	0.00	0.0	36	59	1	0	0	0	0	
4	37.75	2.67	0.0	25	572	0	0	0	1	1	

2. Now build and train a decision tree classifier using `DecisionTreeClassifier()` (manual page) on train_df to predict the `"churndep"` target variable. Make sure to use `criterion='entropy'` when instantiating an instance of `DecisionTreeClassifier()`. For all other settings you should use all of the default options.

(1 Point)

In [5]:
```
# Place your code here
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(criterion='entropy')
dtc.fit(train_X, train_Y)
```

Out[5]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

3. Using the resulting model from 1.2, show a bar plot of feature names and their feature importance (hint: check the attributes of the `DecisionTreeClassifier()` object directly in IPython or check the manual!).

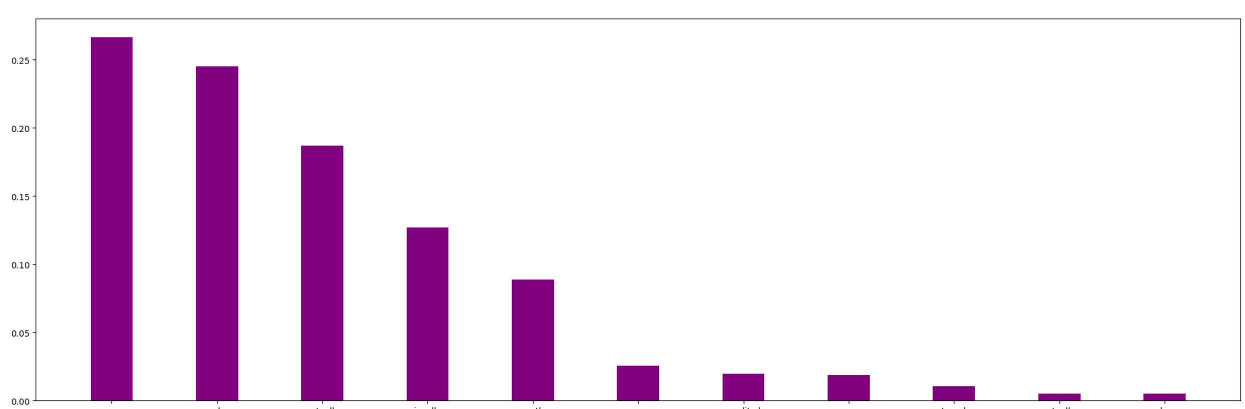
(3 Points)

In [6]:
```
# Place your code here
import matplotlib.pyplot as plt
feature_importance = {}
for importance, name in sorted(zip(dtc.feature_importances_, train_X),reverse=T
    feature_importance[name]=importance

plt.figure(figsize=(25,8))
print(feature_importance)
plt.bar(feature_importance.keys(), feature_importance.values(), color ='purple'
```

```
{'revenue': 0.26670740553298816, 'eqpdays': 0.245072683351932, 'outcalls': 0.1
8701913166536377, 'incalls': 0.12688332321334495, 'months': 0.0886038760442537
8, 'marryyes': 0.025770329551172436, 'creditcd': 0.019889645830896786, 'pcow
n': 0.01882041242271449, 'travel': 0.010798514227739033, 'retcalls': 0.0052604
69301682941, 'webcap': 0.0051742088579116476}
```

Out[6]: <BarContainer object of 11 artists>



4. Is the relationship between the top 3 most important features (as measured here) negative or positive? If your marketing director asked you to explain the top 3 drivers of churn, how would you interpret the relationship between these 3 features and the churn outcome? What "real-life" connection can you draw between each variable and churn?

(2 Points)

In [7]:
```
df_corr = df[["revenue","eqpdays", "outcalls","churndep"]]
df_corr.corr()
```

Out[7]:

	revenue	eqpdays	outcalls	churndep
revenue	1.000000	-0.222074	0.500709	-0.013370
eqpdays	-0.222074	1.000000	-0.244112	0.112821
outcalls	0.500709	-0.244112	1.000000	-0.037071
churndep	-0.013370	0.112821	-0.037071	1.000000

Top 3 drivers of churn are revenue, outcalls and eqpdays of the company revenue <-> churndep -0.013370 negative outcalls<-> churndep -0.037071 negative eqpdays <-> churndep +0.112821 positive

Revenue is negatively correlated that means higher the revenue(Mean monthly revenue in dollars) lower the churndep, similarly outcalls and churndep are also negatively correlated, eqpdays and churndep are positively correlated that is higher eqpdays then higher churndep

Revenue: Revenue is negatively correlated that means higher the revenue lower the churndep It makes sense that if the revenue is high for the company then churndep is low as the clients will trust the company lot more since profits are flowing in Another relation could be that if revenue is flowing in it means clients are flushing the money in so high revenue means low number of clients are leaving and lot of clients are currently paying

Outcalls: Outcalls being negatively correlated also makes sense as it means Mean number of outbound voice calls with the client is lot so client feels a lot more reliable of the company so churndep is lower.

Eqpdays: Number of days the customer has had his/her current equipment : positively correlated which means higher the no of days the customer has their equipment, higher the churndep. May be the equipment becomes faulty after few days and there are no proper customer service because of which the customer tends to leave.

5. Using the classifier built in 1.2, try predicting `"churndep"` on both the train_df and test_df data sets. What is the accuracy on each?

(2 Points)

In [8]:
```
# Place your code here
```

```
# Place your code here
from sklearn.metrics import accuracy_score
predtest_Y = dtc.predict(test_X)
print("Test Accuracy", accuracy_score(test_Y, predtest_Y))

predtrain_Y = dtc.predict(train_X)
print("Train Accuracy", accuracy_score(train_Y, predtrain_Y))
```

```
Test Accuracy 0.5409794468409034
Train Accuracy 0.999780914525367
```

Part 2 - Finding a Good Decision Tree (Total 10 Points)

The default options for your decision tree may not be optimal. We need to analyze whether tuning the parameters can improve the accuracy of the classifier. For the following options `min_samples_split` and `min_samples_leaf`:

1. Generate a list of 10 values of each for the parameters min_samples_split and min_samples_leaf.

(1 Point)

```
min_samples_leaf: int, float, optional (default=1)
The minimum number of samples required to be at a leaf node. A
split point at any depth will only be considered if it leaves
at least min_samples_leaf training samples in each of the left
and right branches. This may have the effect of smoothing the
model, especially in regression.
min_samples_leaf is also used to control over-fitting by
defining that each leaf has more than one element. Thus
ensuring that the tree cannot overfit the training dataset by
creating a bunch of small branches exclusively for one sample
each. In reality, what this is actually doing is simply just
telling the tree that each leaf doesn't have to have an
impurity of 0

min_samples_split: int or float, default=2
The minimum number of samples required to split an internal
node:
If int, then consider min_samples_split as the minimum number.
If float, then min_samples_split is a fraction and
ceil(min_samples_split * n_samples) are the minimum number of
samples for each split.
```

2. Explain in words your reasoning for choosing the above ranges.

```
min_samples_leaf
Chose a range between 100-1750 with a difference of 250 because
```

```
since we have a dataset of 22k and only 2 classes to predict
with only 5 features that contribute to the predictions -
revenue, eqpdays, outcalls, incalls and months - the child
nodes definitely can have a minimum of range 100 and can go
higher as the dataset will be majorly split by 5 features split
- this will also prevent overfitting and generalising the model

min_samples_split
Chose the same range as we don't want the dataset to overfit on
the training sample and having the less than 100 would not only
overfit but will also create a complex tree with multiple nodes
and for 5 contributing features a shallow tree would be
sufficient with multiple samples grouped together in a node
```

Place your response here

3. For each combination of values in 3.1 (there should be 100), build a new classifier and check the classifier's accuracy on the test data. Plot the test set accuracy for these options. Use the values of `min_samples_split` as the x-axis and generate a new series (line) for each of `min_samples_leaf`.

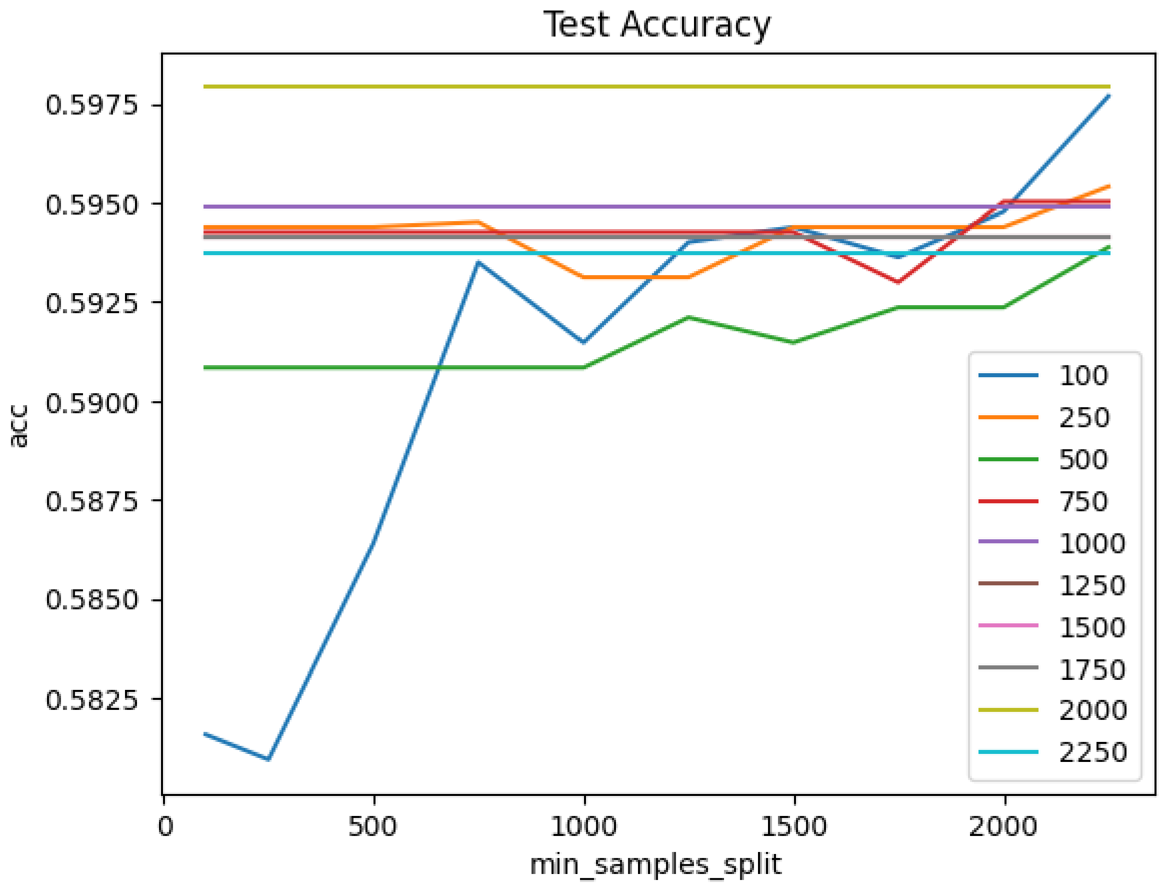
(5 Points)

In [9]:
```
# Place your code here
min_samples_split = [100, 250, 500, 750, 1000, 1250, 1500, 1750, 2000, 2250]
min_samples_leaf =[100, 250, 500, 750, 1000, 1250, 1500, 1750, 2000, 2250]
j=0


for i in range(len(min_samples_leaf)):
    acc_lis = []
    for j in range(len(min_samples_split)):
        dtc = DecisionTreeClassifier(random_state=0, min_samples_split=min_samp
        predtest_Y = dtc.predict(test_X)
        acc = accuracy_score(test_Y, predtest_Y)
        acc_lis.append(acc)

    line = plt.plot(min_samples_split, acc_lis, label = str(min_samples_leaf[i]

plt.xlabel("min_samples_split")
plt.ylabel("acc")
plt.legend()
plt.title("Test Accuracy")
plt.show()
```



4. Which configuration returns the best accuracy? What is this accuracy? (Note, if you don't see much variation in the test set accuracy across values of min_samples_split or min_samples_leaf, try redoing the above steps with a different range of values).

(1 Point)

Best configuration is when min_samples_split is 2000 and min_samples_leaf is 100 and accuracy is around 60.5%

Place your response here

5. If you were working for a marketing department, how would you use your churn production model in a real business environment? Explain why churn prediction might be good for the business and how one might improve churn by using this model.

(2 Points)

I would use this model to periodically check which customer/client might have a tendency to get churned and take precautionary measure beforehand. For example if the model predicts a client/customer is going to leave and then we can always increase outbound and inbound calls as that might make the customer feel they're appreciated and stay with the company for longer or bring in more revenue by transparently showing them how we can improve

I can change my focus to companies that have a greater tendency to drop working with us there by giving them a better service and better deal for them to stay with us longer

Part 3: Model selection with cross-validation (5 points)

In this part, we will focus on cross-validation to find a good value for parameter `max_depth`.

1. Write a cross-validation function that does the following:

- Takes as inputs a dataset, a label name, # of splits/folds (`k`), and a sequence of values for the maximum depth of the tree (`max_depth`).
- Shuffles the data.
- Splits the data into `k` folds according to the cross-validation logic
- Performs two loops
  - Outer Loop: `for each f in range(k)`:
    - Inner Loop: `for each value in max_depth_sequence`:
      - Trains a Decision Tree on the training split with the `max_depth=value` (USE criterion='entropy' BUT DO NOT ALTER THE OTHER PARAMETERS)
      - Computes accuracy_value_f on test split
      - Stores accuracy_value_f in a dictionary of values
- Returns a dictionary, where each key-value pair is: `value: [accuracy_value_1,...,accuracy_value_k]`

(2 Points)

In [10]:
```
from random import randrange

def xValDecisionTree(dataset, label, k, max_depth_sequence):
    dataset = dataset.sample(frac=1).reset_index(drop=True)
    df_split = np.array_split(dataset, k)
    train_split = pd.DataFrame()
    acc_dict = {}
    for i in range(0,k):
        test_split = pd.DataFrame(df_split[i])
        for j in range(k):
            if i!= j:
                train_split = pd.concat([pd.DataFrame(df_split[j]), train_split
        acc_lis = []
        depth_lis = []

        train_X = train_split.loc[:, train_split.columns != label]
        train_Y = train_split[label]


        test_X = test_split.loc[:, test_split.columns != label]
        test_Y = test_split[label]


        for value in max_depth_sequence:
```

```
            dtc = DecisionTreeClassifier(random_state=0, max_depth=value,  crit
            predtest_Y = dtc.predict(test_X)
            acc = accuracy_score(test_Y, predtest_Y)
            acc_lis.append(acc)
            depth_lis.append(value)
            if value not in acc_dict.keys():
                acc_dict[value] = []
            acc_dict[value].append(acc)
        plt.show()

    return acc_dict
```

2. Using the function written above, do the following:

- Generate a sequence `max_depth_sequence = [None, 2, 4, 8, 16, 32, 128, 256, 512]` (Note that None is the default value for this parameter).

2. Call accs = xValDecisionTree(dataset, 'churndep', 10, `max_depth_sequence`)
3. For each value in accs.keys(), calculate mean(accs[value]). What value is associated with the highest accuracy mean?
4. For each value in accs.keys(), calculate the ranges mean(accs[value]) +/- std(accs[value]). Do the ranges associated with the value that has the highest mean(accs[value]) overlap with ranges for other values? What may this suggest and what are the limitations of a standard deviation based analysis in this scenario?
5. Which depth value would you pick, if any, and why?

(3 Points)

In [11]:
```
# Place your code here
#2.
max_depth_sequence = [None, 2, 4, 8, 16, 32, 128, 256, 512]
accs_dict = xValDecisionTree(df, "churndep", 10,  max_depth_sequence)
pprint(accs_dict)
```

```
{None: [0.5261044176706827,
        1.0,
        0.9997489959839357,
        0.9997489329651017,
        1.0,
        0.9994978659302034,
        0.9992467988953051,
        1.0,
        0.9997489329651017,
        0.9994978659302034],
 2: [0.5911144578313253,
     0.5753012048192772,
     0.5700301204819277,
     0.5761988450916394,
     0.5842329902083856,
     0.5819733868943008,
     0.5721817725332664,
     0.5902585990459452,
     0.6010544815465729,
     0.5824755209640974],
 4: [0.6019076305220884,
     0.5896084337349398,
     0.5868473895582329,
     0.5877479286969621,
     0.5912628671855386,
     0.5972884760230982,
     0.5937735375345217,
     0.6073311574190309,
     0.6118503640472006,
     0.5937735375345217],
 8: [0.5971385542168675,
     0.5946285140562249,
     0.6069277108433735,
     0.5980416771277931,
     0.6015566156163695,
     0.6073311574190309,
     0.5997991463720813,
     0.62239517951293,
     0.6201355761988451,
     0.6108460959076073],
 16: [0.5740461847389559,
      0.6646586345381527,
      0.6814759036144579,
      0.6721064524227969,
      0.6829023349234246,
      0.6889279437609842,
      0.7034898317850866,
      0.7122771780065278,
      0.707004770273663,
      0.6969620888777304],
 32: [0.5306224899598394,
      0.9299698795180723,
      0.9362449799196787,
      0.9357268390660306,
      0.9440120512176752,
      0.9467737886015566,
      0.9452673863921667,
```

```
      0.9485312578458448,
      0.9492844589505398,
      0.94627165453176],
 128: [0.5261044176706827,
       1.0,
       0.9997489959839357,
       0.9997489329651017,
       1.0,
       0.9994978659302034,
       0.9992467988953051,
       1.0,
       0.9997489329651017,
       0.9994978659302034],
 256: [0.5261044176706827,
       1.0,
       0.9997489959839357,
       0.9997489329651017,
       1.0,
       0.9994978659302034,
       0.9992467988953051,
       1.0,
       0.9997489329651017,
       0.9994978659302034],
 512: [0.5261044176706827,
       1.0,
       0.9997489959839357,
       0.9997489329651017,
       1.0,
       0.9994978659302034,
       0.9992467988953051,
       1.0,
       0.9997489329651017,
       0.9994978659302034]}
```

In [12]:
```
accs_df = pd.DataFrame.from_dict(accs_dict)
accs_df
```

Out[12]:

	NaN	2.0	4.0	8.0	16.0	32.0	128.0	256.0	512
0	0.526104	0.591114	0.601908	0.597139	0.574046	0.530622	0.526104	0.526104	0.52610
1	1.000000	0.575301	0.589608	0.594629	0.664659	0.929970	1.000000	1.000000	1.00000
2	0.999749	0.570030	0.586847	0.606928	0.681476	0.936245	0.999749	0.999749	0.99974
3	0.999749	0.576199	0.587748	0.598042	0.672106	0.935727	0.999749	0.999749	0.99974
4	1.000000	0.584233	0.591263	0.601557	0.682902	0.944012	1.000000	1.000000	1.00000
5	0.999498	0.581973	0.597288	0.607331	0.688928	0.946774	0.999498	0.999498	0.99949
6	0.999247	0.572182	0.593774	0.599799	0.703490	0.945267	0.999247	0.999247	0.99924
7	1.000000	0.590259	0.607331	0.622395	0.712277	0.948531	1.000000	1.000000	1.00000
8	0.999749	0.601054	0.611850	0.620136	0.707005	0.949284	0.999749	0.999749	0.99974
9	0.999498	0.582476	0.593774	0.610846	0.696962	0.946272	0.999498	0.999498	0.99949

3. The maximum accuracy in the val set is 0.952359 when max_depth = None, 128, 256,

512

In [13]:
```
#3
describe_df = accs_df.describe()
describe_df.loc['mean-std'] = describe_df.loc['mean'] - describe_df.loc['std']
describe_df.loc['mean+std'] = describe_df.loc['mean'] + describe_df.loc['std']
describe_df
```

Out[13]:

	NaN	2.0	4.0	8.0	16.0	32.0	128.0	
count	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.0
mean	0.952359	0.582482	0.596139	0.605880	0.678385	0.901270	0.952359	0.9
std	0.149771	0.009635	0.008440	0.009568	0.039719	0.130389	0.149771	0.
min	0.526104	0.570030	0.586847	0.594629	0.574046	0.530622	0.526104	0.5
25%	0.999498	0.575526	0.590022	0.598481	0.674449	0.935856	0.999498	0.9
50%	0.999749	0.582224	0.593774	0.604242	0.685915	0.944640	0.999749	0.9
75%	0.999937	0.588752	0.600753	0.609967	0.701858	0.946648	0.999937	0.9
max	1.000000	0.601054	0.611850	0.622395	0.712277	0.949284	1.000000	1.0
mean-std	0.802588	0.572847	0.587699	0.596312	0.638666	0.770882	0.802588	0.8
mean+std	1.102130	0.592117	0.604579	0.615448	0.718105	1.031659	1.102130	1.

4. The range(mean-std and mean+std) is teh same for depth = None, 128, 256, 512 range: (0.803 - 1.102)

Standard deviation here shows how the data in folds are distributed that is if the std of accuracy is low then it means that the data in the trainset and val set have been learnt uniformly and there is no overfit. If the std is high which means that the training has been overfit on the train fold and in the val fold the accuracy is not being able to replicate.

Like example take max_depth 16 and when fold 0 was val fold; accuracy was 0.59 whereas when fold 1 was used as val fold accuracy jumped to 0.67 which means the data in val fold 0 is a little different from the data in the training that implies for max_depth 16 the model is overfitting to the training folds and is extremely dependant on it because of which val fold accuracy starts fluctuating

5. The range(mean - std to mean + std) is the same for depth = None, 128, 256 and 512 its the same range that is 0.803 - 1.102

It means the data is overfitting to the train data so the val set accuracy is extremely high So, i'll probably take the next best accuracy that is with value 16.0 or 8.0 as max_depth_sequence I am inclining more towards 8 because the std is less than 16 which means that accuracy of a set similar to trainset will be in the smaller range from 0.596 - 0.62 unlike 16 which has a larger range which means features learnt with max_depth =16 did not help in predicting val fold - overfit to the training fold.

In [14]:
```
dtc = DecisionTreeClassifier(random_state=0, max_depth=None,  criterion='entrop
predtest_Y = dtc.predict(test_X)
acc = accuracy_score(test_Y, predtest_Y)
print("For max_depth = None Acc = ",acc)


dtc = DecisionTreeClassifier(random_state=0, max_depth=256,  criterion='entropy
predtest_Y = dtc.predict(test_X)
acc = accuracy_score(test_Y, predtest_Y)
print("For max_depth = 256 Acc = ",acc)

dtc = DecisionTreeClassifier(random_state=0, max_depth=512,  criterion='entropy
predtest_Y = dtc.predict(test_X)
acc = accuracy_score(test_Y, predtest_Y)
print("For max_depth = 512 Acc = ",acc)
```

```
For max_depth = None Acc =  0.5385688911443796
For max_depth = 256 Acc =  0.5385688911443796
For max_depth = 512 Acc =  0.5385688911443796
```

This shows how the test data is not performing that great on the best performing model in the train data as those iteration are over fitting, whereas the one with max_depth of 16 has generalised well so performs better than max_depth = None, 128, 256, 512

and highest for max_depth = 8

In [15]:
```
dtc = DecisionTreeClassifier(random_state=0, max_depth=16,  criterion='entropy'
predtest_Y = dtc.predict(test_X)
acc = accuracy_score(test_Y, predtest_Y)
print("For max_depth = 16 Acc = ",acc)
```

```
For max_depth = 16 Acc =  0.5749809692971327
```

In [16]:
```
dtc = DecisionTreeClassifier(random_state=0, max_depth=32,  criterion='entropy'
predtest_Y = dtc.predict(test_X)
acc = accuracy_score(test_Y, predtest_Y)
print("For max_depth = 32 Acc = ",acc)
```

```
For max_depth = 32 Acc =  0.5383151484394824
```

In [17]:
```
dtc = DecisionTreeClassifier(random_state=0, max_depth=8,  criterion='entropy'
predtest_Y = dtc.predict(test_X)
acc = accuracy_score(test_Y, predtest_Y)
print("For max_depth = 8 Acc = ",acc)
```

```
For max_depth = 8 Acc =  0.5912205024105557
```

Part 4: Boosting (5 Points)

Now, as we covered in class, ensemble methods are often used to improve performance.

1. Implement the boosting algorithm: XGBoost for the same `cell2cell_data.csv` task as above. You will have to select how to tune hyperparameters. Besides depth, which other hyperparametrs do you optimize for? (2 points)

XGBoost hyperparameters

General Parameters
booster
gbtree defaut which is okay as it is used for tree based xgboost

Booster Parameters
eta
It is the step size shrinkage used in update to prevent overfitting.
Range : [0,1] Typical final values : 0.01-0.2.\n
Default : 0

gamma
A node is split only when the resulting split gives a positive reduction in the loss function. The larger gamma is, the more conservative the algorithm will be.
Range: [0,∞]
Default : 0

max_depth - used 3 because mostly there are 3 major features that is eqpdays, revenue, outcalls
It is used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
range: [0,∞] (0 is only accepted in lossguided growing policy when tree_method is set as hist.
default =6

min_child_weight - for overfitting we have modified tree depth etc
It defines the minimum sum of weights of all observations required in a child.
range: [0,∞]
It is used to control over-fitting.
Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.
default=1

max_delta_step - churndep is not biased(0: 15991 1: 15951) so not using this
In maximum delta step we allow each tree’s weight estimation to be.
Usually this parameter is not needed, but it might help in logistic regression when class is extremely imbalanced.
Set it to value of 1-10 might help control the update.
range: [0,∞]
default=0

subsample - Want to give this default because this is too much rules on the tree
It denotes the fraction of observations to be randomly samples for each tree.
Subsample ratio of the training instances.
Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees. - This will prevent overfitting.
Typical values: 0.5-1
range: (0,1]

lambda - 1 is good enough
L2 regularization term on weights (analogous to Ridge regression).
This is used to handle the regularization part of XGBoost.
Increasing this value will make model more conservative.
default=1

alpha - No need this because less features and we don't need feature sleection here
L1 regularization term on weights (analogous to Lasso regression).
It can be used in case of very high dimensionality so that the algorithm runs faster when implemented.
Increasing this value will make model more conservative.
default=0

tree_method - auto takes care of which one to choose based on dataset size so no need of change
Choices: auto, exact, approx, hist, gpu_hist
auto: Use heuristic to choose the fastest method.
For small to medium dataset, exact greedy (exact) will be used.
For very large dataset, approximate algorithm (approx) will be chosen.
Because old behavior is always use exact greedy in single machine, user will get a message when approximate algorithm is chosen to notify this choice.
exact: Exact greedy algorithm.
approx: Approximate greedy algorithm using quantile sketch and gradient histogram.
hist: Fast histogram optimized approximate greedy algorithm. It uses some performance improvements such as bins caching.
gpu_hist: GPU implementation of hist algorithm.
default= auto

scale_pos_weight - No need as classes are balanced for us
It controls the balance of positive and negative weights,
It is useful for imbalanced classes.
A value greater than 0 should be used in case of high class

```
imbalance as it helps in faster convergence.
A typical value to consider: sum(negative instances) /
sum(positive instances).
default=1


max_leaves  - too much interference to the tree so no need
Maximum number of nodes to be added.
Only relevant when grow_policy=lossguide is set.


Learning Task Parameters

objective  - binary:logistic because this is binary
classification
reg:logistic : logistic regression
binary:logistic : logistic regression for binary
classification, output probability
binary:logitraw: logistic regression for binary classification,
output score before logistic transformation
binary:hinge : hinge loss for binary classification. This makes
predictions of 0 or 1, rather than producing probabilities.
multi:softmax : set XGBoost to do multiclass classification
using the softmax objective, you also need to set
num_class(number of classes)
multi:softprob : same as softmax, but output a vector of ndata
nclass, which can be further reshaped to ndata nclass matrix.
The result contains predicted probability of each data point
belonging to each class.
default=reg:squarederror

eval_metric - default metrics makes sense
The metric to be used for validation data.
The default values are rmse for regression, error for
classification and mean average precision for ranking.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Place your answer here regarding hyperparameters.

In [18]:
```
import xgboost as xgb
xgb_model = xgb.XGBClassifier()
xgb_model.fit(train_X, train_Y)
y_pred = xgb_model.predict(test_X)
```

```
acc = accuracy_score(test_Y, y_pred)
print(acc)
```

0.5825932504440497

Method 1 : Manual change based on understanding

In [19]:
```
import xgboost as xgb
xgb_model = xgb.XGBClassifier(booster = 'dart', eta = 0.3, max_depth = 3, objec
eval_set = [(train_X, train_Y), (test_X, test_Y)]
xgb_model.fit(train_X, train_Y, eval_set=eval_set, eval_metric=["error", "loglo
y_pred = xgb_model.predict(test_X)
acc = accuracy_score(test_Y, y_pred)
print(acc)

import matplotlib.pyplot as plt
results = xgb_model.evals_result()
epochs = len(results['validation_0']['error'])
x_axis = range(0, epochs)
# plot log loss
fig, ax = plt.subplots()
ax.plot(x_axis, results['validation_0']['logloss'], label='Train')
ax.plot(x_axis, results['validation_1']['logloss'], label='Test')
ax.legend()
```

```
[0]	validation_0-error:0.40762	validation_0-logloss:0.68130	valida
tion_1-error:0.40624	validation_1-logloss:0.68073
[1]	validation_0-error:0.40537	validation_0-logloss:0.67523	valida
tion_1-error:0.40358	validation_1-logloss:0.67434
[2]	validation_0-error:0.41235	validation_0-logloss:0.67201	valida
tion_1-error:0.41373	validation_1-logloss:0.67128
[3]	validation_0-error:0.40446	validation_0-logloss:0.66949	valida
tion_1-error:0.40396	validation_1-logloss:0.66912
[4]	validation_0-error:0.41166	validation_0-logloss:0.66813	valida
tion_1-error:0.41474	validation_1-logloss:0.66821
[5]	validation_0-error:0.40371	validation_0-logloss:0.66641	valida
tion_1-error:0.40472	validation_1-logloss:0.66672
[6]	validation_0-error:0.40262	validation_0-logloss:0.66548	valida
tion_1-error:0.40256	validation_1-logloss:0.66606
[7]	validation_0-error:0.40080	validation_0-logloss:0.66454	valida
tion_1-error:0.40129	validation_1-logloss:0.66555
[8]	validation_0-error:0.40115	validation_0-logloss:0.66347	valida
tion_1-error:0.40269	validation_1-logloss:0.66443
[9]	validation_0-error:0.39902	validation_0-logloss:0.66290	valida
tion_1-error:0.40180	validation_1-logloss:0.66442
[10]	validation_0-error:0.39911	validation_0-logloss:0.66211	valida
tion_1-error:0.40256	validation_1-logloss:0.66454
[11]	validation_0-error:0.39833	validation_0-logloss:0.66143	valida
tion_1-error:0.40244	validation_1-logloss:0.66435
[12]	validation_0-error:0.39733	validation_0-logloss:0.66096	valida
tion_1-error:0.40244	validation_1-logloss:0.66420
[13]	validation_0-error:0.39717	validation_0-logloss:0.66067	valida
tion_1-error:0.40269	validation_1-logloss:0.66411
[14]	validation_0-error:0.39680	validation_0-logloss:0.66010	valida
tion_1-error:0.40256	validation_1-logloss:0.66374
[15]	validation_0-error:0.39611	validation_0-logloss:0.65972	valida
tion_1-error:0.40307	validation_1-logloss:0.66374
[16]	validation_0-error:0.39579	validation_0-logloss:0.65952	valida
tion_1-error:0.40282	validation_1-logloss:0.66375
[17]	validation_0-error:0.39645	validation_0-logloss:0.65929	valida
tion_1-error:0.40358	validation_1-logloss:0.66386
```

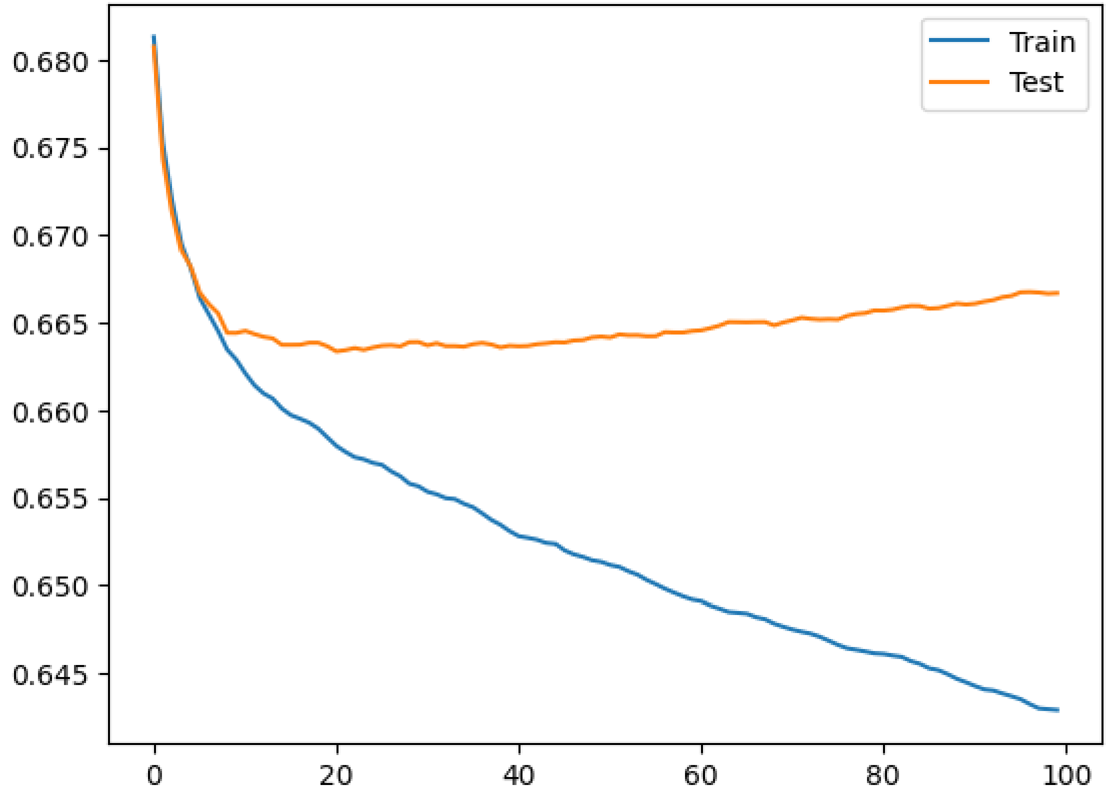
```
/Users/yamini/Library/Python/3.8/lib/python/site-packages/xgboost/sklearn.py:7
93: UserWarning: `eval_metric` in `fit` method is deprecated for better compat
ibility with scikit-learn, use `eval_metric` in constructor or`set_params` ins
tead.
  warnings.warn(
```

```
[18]	validation_0-error:0.39586	validation_0-logloss:0.65895	valida
tion_1-error:0.40434	validation_1-logloss:0.66386
[19]	validation_0-error:0.39526	validation_0-logloss:0.65845	valida
tion_1-error:0.40332	validation_1-logloss:0.66366
[20]	validation_0-error:0.39470	validation_0-logloss:0.65796	valida
tion_1-error:0.40409	validation_1-logloss:0.66337
[21]	validation_0-error:0.39442	validation_0-logloss:0.65762	valida
tion_1-error:0.40383	validation_1-logloss:0.66344
[22]	validation_0-error:0.39282	validation_0-logloss:0.65732	valida
tion_1-error:0.40573	validation_1-logloss:0.66356
[23]	validation_0-error:0.39257	validation_0-logloss:0.65720	valida
tion_1-error:0.40510	validation_1-logloss:0.66345
[24]	validation_0-error:0.39216	validation_0-logloss:0.65700	valida
tion_1-error:0.40459	validation_1-logloss:0.66359
[25]	validation_0-error:0.39223	validation_0-logloss:0.65688	valida
tion_1-error:0.40459	validation_1-logloss:0.66369
[26]	validation_0-error:0.39132	validation_0-logloss:0.65651	valida
tion_1-error:0.40497	validation_1-logloss:0.66372
[27]	validation_0-error:0.39119	validation_0-logloss:0.65624	valida
tion_1-error:0.40459	validation_1-logloss:0.66365
[28]	validation_0-error:0.39091	validation_0-logloss:0.65581	valida
tion_1-error:0.40472	validation_1-logloss:0.66389
[29]	validation_0-error:0.39054	validation_0-logloss:0.65566	valida
tion_1-error:0.40472	validation_1-logloss:0.66390
[30]	validation_0-error:0.39013	validation_0-logloss:0.65535	valida
tion_1-error:0.40396	validation_1-logloss:0.66371
[31]	validation_0-error:0.38991	validation_0-logloss:0.65520	valida
tion_1-error:0.40434	validation_1-logloss:0.66383
[32]	validation_0-error:0.38913	validation_0-logloss:0.65498	valida
tion_1-error:0.40320	validation_1-logloss:0.66367
[33]	validation_0-error:0.38919	validation_0-logloss:0.65493	valida
tion_1-error:0.40320	validation_1-logloss:0.66367
[34]	validation_0-error:0.38825	validation_0-logloss:0.65466	valida
tion_1-error:0.40472	validation_1-logloss:0.66363
[35]	validation_0-error:0.38834	validation_0-logloss:0.65446	valida
tion_1-error:0.40421	validation_1-logloss:0.66378
[36]	validation_0-error:0.38737	validation_0-logloss:0.65411	valida
tion_1-error:0.40523	validation_1-logloss:0.66385
[37]	validation_0-error:0.38656	validation_0-logloss:0.65374	valida
tion_1-error:0.40409	validation_1-logloss:0.66376
[38]	validation_0-error:0.38625	validation_0-logloss:0.65346	valida
tion_1-error:0.40320	validation_1-logloss:0.66359
[39]	validation_0-error:0.38500	validation_0-logloss:0.65310	valida
tion_1-error:0.40294	validation_1-logloss:0.66369
[40]	validation_0-error:0.38456	validation_0-logloss:0.65281	valida
tion_1-error:0.40218	validation_1-logloss:0.66366
[41]	validation_0-error:0.38468	validation_0-logloss:0.65272	valida
tion_1-error:0.40244	validation_1-logloss:0.66368
[42]	validation_0-error:0.38465	validation_0-logloss:0.65261	valida
tion_1-error:0.40256	validation_1-logloss:0.66378
[43]	validation_0-error:0.38481	validation_0-logloss:0.65243	valida
tion_1-error:0.40231	validation_1-logloss:0.66383
[44]	validation_0-error:0.38471	validation_0-logloss:0.65237	valida
tion_1-error:0.40231	validation_1-logloss:0.66389
[45]	validation_0-error:0.38418	validation_0-logloss:0.65200	valida
tion_1-error:0.40091	validation_1-logloss:0.66388
[46]	validation_0-error:0.38390	validation_0-logloss:0.65177	valida
```

```
tion_1-error:0.40244	validation_1-logloss:0.66399
[47]	validation_0-error:0.38378	validation_0-logloss:0.65163	valida
tion_1-error:0.40231	validation_1-logloss:0.66401
[48]	validation_0-error:0.38343	validation_0-logloss:0.65144	valida
tion_1-error:0.40332	validation_1-logloss:0.66417
[49]	validation_0-error:0.38346	validation_0-logloss:0.65135	valida
tion_1-error:0.40332	validation_1-logloss:0.66421
[50]	validation_0-error:0.38321	validation_0-logloss:0.65116	valida
tion_1-error:0.40282	validation_1-logloss:0.66416
[51]	validation_0-error:0.38324	validation_0-logloss:0.65105	valida
tion_1-error:0.40269	validation_1-logloss:0.66434
[52]	validation_0-error:0.38271	validation_0-logloss:0.65081	valida
tion_1-error:0.40269	validation_1-logloss:0.66429
[53]	validation_0-error:0.38290	validation_0-logloss:0.65061	valida
tion_1-error:0.40345	validation_1-logloss:0.66429
[54]	validation_0-error:0.38168	validation_0-logloss:0.65031	valida
tion_1-error:0.40485	validation_1-logloss:0.66423
[55]	validation_0-error:0.38111	validation_0-logloss:0.65007	valida
tion_1-error:0.40447	validation_1-logloss:0.66424
[56]	validation_0-error:0.38065	validation_0-logloss:0.64981	valida
tion_1-error:0.40396	validation_1-logloss:0.66447
[57]	validation_0-error:0.38011	validation_0-logloss:0.64959	valida
tion_1-error:0.40332	validation_1-logloss:0.66445
[58]	validation_0-error:0.38011	validation_0-logloss:0.64938	valida
tion_1-error:0.40269	validation_1-logloss:0.66445
[59]	validation_0-error:0.38014	validation_0-logloss:0.64920	valida
tion_1-error:0.40193	validation_1-logloss:0.66453
[60]	validation_0-error:0.38002	validation_0-logloss:0.64911	valida
tion_1-error:0.40180	validation_1-logloss:0.66457
[61]	validation_0-error:0.37946	validation_0-logloss:0.64886	valida
tion_1-error:0.40269	validation_1-logloss:0.66470
[62]	validation_0-error:0.37942	validation_0-logloss:0.64869	valida
tion_1-error:0.40256	validation_1-logloss:0.66484
[63]	validation_0-error:0.37917	validation_0-logloss:0.64850	valida
tion_1-error:0.40218	validation_1-logloss:0.66504
[64]	validation_0-error:0.37939	validation_0-logloss:0.64845	valida
tion_1-error:0.40218	validation_1-logloss:0.66504
[65]	validation_0-error:0.37933	validation_0-logloss:0.64840	valida
tion_1-error:0.40218	validation_1-logloss:0.66503
[66]	validation_0-error:0.37889	validation_0-logloss:0.64821	valida
tion_1-error:0.40218	validation_1-logloss:0.66504
[67]	validation_0-error:0.37864	validation_0-logloss:0.64809	valida
tion_1-error:0.40231	validation_1-logloss:0.66504
[68]	validation_0-error:0.37855	validation_0-logloss:0.64783	valida
tion_1-error:0.40307	validation_1-logloss:0.66485
[69]	validation_0-error:0.37855	validation_0-logloss:0.64766	valida
tion_1-error:0.40332	validation_1-logloss:0.66502
[70]	validation_0-error:0.37824	validation_0-logloss:0.64750	valida
tion_1-error:0.40332	validation_1-logloss:0.66514
[71]	validation_0-error:0.37799	validation_0-logloss:0.64738	valida
tion_1-error:0.40332	validation_1-logloss:0.66529
[72]	validation_0-error:0.37783	validation_0-logloss:0.64727	valida
tion_1-error:0.40332	validation_1-logloss:0.66523
[73]	validation_0-error:0.37767	validation_0-logloss:0.64709	valida
tion_1-error:0.40358	validation_1-logloss:0.66519
[74]	validation_0-error:0.37695	validation_0-logloss:0.64686	valida
tion_1-error:0.40206	validation_1-logloss:0.66521
```

```
[75]	validation_0-error:0.37695	validation_0-logloss:0.64663	valida
tion_1-error:0.40193	validation_1-logloss:0.66519
[76]	validation_0-error:0.37664	validation_0-logloss:0.64644	valida
tion_1-error:0.40231	validation_1-logloss:0.66540
[77]	validation_0-error:0.37661	validation_0-logloss:0.64635	valida
tion_1-error:0.40231	validation_1-logloss:0.66551
[78]	validation_0-error:0.37658	validation_0-logloss:0.64625	valida
tion_1-error:0.40231	validation_1-logloss:0.66555
[79]	validation_0-error:0.37651	validation_0-logloss:0.64614	valida
tion_1-error:0.40244	validation_1-logloss:0.66571
[80]	validation_0-error:0.37651	validation_0-logloss:0.64611	valida
tion_1-error:0.40231	validation_1-logloss:0.66570
[81]	validation_0-error:0.37626	validation_0-logloss:0.64602	valida
tion_1-error:0.40231	validation_1-logloss:0.66575
[82]	validation_0-error:0.37623	validation_0-logloss:0.64594	valida
tion_1-error:0.40269	validation_1-logloss:0.66588
[83]	validation_0-error:0.37636	validation_0-logloss:0.64570	valida
tion_1-error:0.40421	validation_1-logloss:0.66597
[84]	validation_0-error:0.37636	validation_0-logloss:0.64554	valida
tion_1-error:0.40345	validation_1-logloss:0.66596
[85]	validation_0-error:0.37617	validation_0-logloss:0.64529	valida
tion_1-error:0.40332	validation_1-logloss:0.66581
[86]	validation_0-error:0.37604	validation_0-logloss:0.64518	valida
tion_1-error:0.40345	validation_1-logloss:0.66584
[87]	validation_0-error:0.37592	validation_0-logloss:0.64495	valida
tion_1-error:0.40282	validation_1-logloss:0.66597
[88]	validation_0-error:0.37598	validation_0-logloss:0.64469	valida
tion_1-error:0.40320	validation_1-logloss:0.66610
[89]	validation_0-error:0.37561	validation_0-logloss:0.64449	valida
tion_1-error:0.40294	validation_1-logloss:0.66604
[90]	validation_0-error:0.37476	validation_0-logloss:0.64428	valida
tion_1-error:0.40396	validation_1-logloss:0.66609
[91]	validation_0-error:0.37460	validation_0-logloss:0.64408	valida
tion_1-error:0.40396	validation_1-logloss:0.66621
[92]	validation_0-error:0.37454	validation_0-logloss:0.64402	valida
tion_1-error:0.40409	validation_1-logloss:0.66630
[93]	validation_0-error:0.37435	validation_0-logloss:0.64385	valida
tion_1-error:0.40396	validation_1-logloss:0.66647
[94]	validation_0-error:0.37420	validation_0-logloss:0.64370	valida
tion_1-error:0.40370	validation_1-logloss:0.66654
[95]	validation_0-error:0.37363	validation_0-logloss:0.64352	valida
tion_1-error:0.40447	validation_1-logloss:0.66674
[96]	validation_0-error:0.37360	validation_0-logloss:0.64324	valida
tion_1-error:0.40447	validation_1-logloss:0.66676
[97]	validation_0-error:0.37307	validation_0-logloss:0.64300	valida
tion_1-error:0.40447	validation_1-logloss:0.66673
[98]	validation_0-error:0.37301	validation_0-logloss:0.64295	valida
tion_1-error:0.40434	validation_1-logloss:0.66667
[99]	validation_0-error:0.37307	validation_0-logloss:0.64291	valida
tion_1-error:0.40434	validation_1-logloss:0.66670
0.5956609997462573
```

Out[19]:
```
<matplotlib.legend.Legend at 0x1588730d0>
```



Testing log loss is increasing in test after sometime, which might be because of overfitting on the training data and so we'll reduce the max_depth from 3 to 2

In [26]:
```
import xgboost as xgb
xgb_model = xgb.XGBClassifier(booster = 'dart', eta = 0.3, max_depth = 2, objec
eval_set = [(train_X, train_Y), (test_X, test_Y)]
xgb_model.fit(train_X, train_Y, eval_set=eval_set, eval_metric=["error", "loglo
y_pred = xgb_model.predict(test_X)
acc = accuracy_score(test_Y, y_pred)
print(acc)

import matplotlib.pyplot as plt
results = xgb_model.evals_result()
epochs = len(results['validation_0']['error'])
x_axis = range(0, epochs)
# plot log loss
fig, ax = plt.subplots()
ax.plot(x_axis, results['validation_0']['logloss'], label='Train')
ax.plot(x_axis, results['validation_1']['logloss'], label='Test')
ax.legend()
```

```
[0]	validation_0-error:0.41748	validation_0-logloss:0.68375	valida
tion_1-error:0.41804	validation_1-logloss:0.68314
[1]	validation_0-error:0.41304	validation_0-logloss:0.67849	valida
tion_1-error:0.41322	validation_1-logloss:0.67810
[2]	validation_0-error:0.41304	validation_0-logloss:0.67501	valida
tion_1-error:0.41335	validation_1-logloss:0.67430
[3]	validation_0-error:0.41263	validation_0-logloss:0.67296	valida
tion_1-error:0.41284	validation_1-logloss:0.67247
[4]	validation_0-error:0.41263	validation_0-logloss:0.67171	valida
tion_1-error:0.41284	validation_1-logloss:0.67123
[5]	validation_0-error:0.41251	validation_0-logloss:0.67023	valida
tion_1-error:0.41373	validation_1-logloss:0.66938
[6]	validation_0-error:0.41216	validation_0-logloss:0.66935	valida
tion_1-error:0.41373	validation_1-logloss:0.66884
[7]	validation_0-error:0.41194	validation_0-logloss:0.66868	valida
tion_1-error:0.41360	validation_1-logloss:0.66809
[8]	validation_0-error:0.41197	validation_0-logloss:0.66822	valida
tion_1-error:0.41360	validation_1-logloss:0.66804
[9]	validation_0-error:0.41166	validation_0-logloss:0.66766	valida
tion_1-error:0.41271	validation_1-logloss:0.66778
[10]	validation_0-error:0.41110	validation_0-logloss:0.66711	valida
tion_1-error:0.41284	validation_1-logloss:0.66729
[11]	validation_0-error:0.40891	validation_0-logloss:0.66631	valida
tion_1-error:0.40992	validation_1-logloss:0.66670
[12]	validation_0-error:0.40612	validation_0-logloss:0.66585	valida
tion_1-error:0.40713	validation_1-logloss:0.66653
[13]	validation_0-error:0.40603	validation_0-logloss:0.66554	valida
tion_1-error:0.40764	validation_1-logloss:0.66634
[14]	validation_0-error:0.40662	validation_0-logloss:0.66526	valida
tion_1-error:0.40865	validation_1-logloss:0.66630
[15]	validation_0-error:0.40506	validation_0-logloss:0.66494	valida
tion_1-error:0.40764	validation_1-logloss:0.66622
```

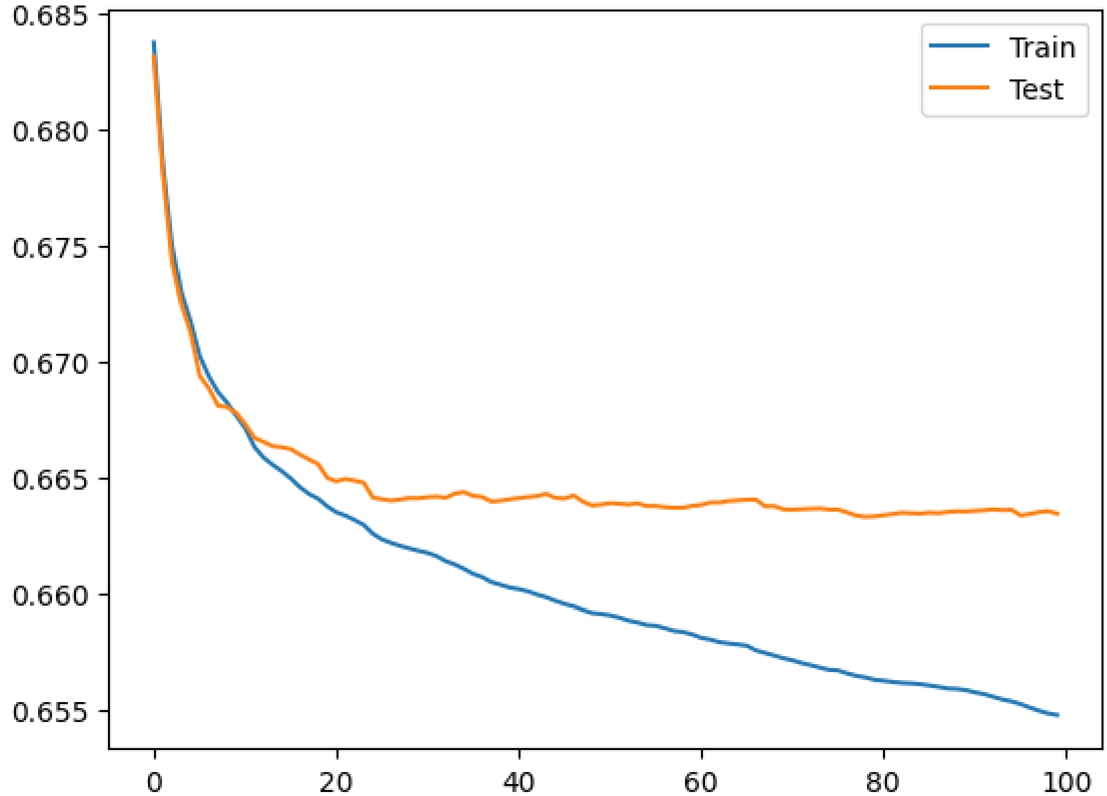
```
/Users/yamini/Library/Python/3.8/lib/python/site-packages/xgboost/sklearn.py:7
93: UserWarning: `eval_metric` in `fit` method is deprecated for better compat
ibility with scikit-learn, use `eval_metric` in constructor or`set_params` ins
tead.
  warnings.warn(
```

```
[16]	validation_0-error:0.40434	validation_0-logloss:0.66457	valida
tion_1-error:0.40726	validation_1-logloss:0.66598
[17]	validation_0-error:0.40393	validation_0-logloss:0.66428	valida
tion_1-error:0.40662	validation_1-logloss:0.66577
[18]	validation_0-error:0.40428	validation_0-logloss:0.66407	valida
tion_1-error:0.40675	validation_1-logloss:0.66557
[19]	validation_0-error:0.40246	validation_0-logloss:0.66374	valida
tion_1-error:0.40497	validation_1-logloss:0.66497
[20]	validation_0-error:0.40215	validation_0-logloss:0.66349	valida
tion_1-error:0.40345	validation_1-logloss:0.66483
[21]	validation_0-error:0.40215	validation_0-logloss:0.66334	valida
tion_1-error:0.40332	validation_1-logloss:0.66493
[22]	validation_0-error:0.40218	validation_0-logloss:0.66315	valida
tion_1-error:0.40472	validation_1-logloss:0.66486
[23]	validation_0-error:0.40233	validation_0-logloss:0.66294	valida
tion_1-error:0.40612	validation_1-logloss:0.66476
[24]	validation_0-error:0.40202	validation_0-logloss:0.66256	valida
tion_1-error:0.40345	validation_1-logloss:0.66412
[25]	validation_0-error:0.40190	validation_0-logloss:0.66232	valida
tion_1-error:0.40370	validation_1-logloss:0.66405
[26]	validation_0-error:0.40140	validation_0-logloss:0.66217	valida
tion_1-error:0.40421	validation_1-logloss:0.66399
[27]	validation_0-error:0.40118	validation_0-logloss:0.66204	valida
tion_1-error:0.40358	validation_1-logloss:0.66404
[28]	validation_0-error:0.40121	validation_0-logloss:0.66193	valida
tion_1-error:0.40447	validation_1-logloss:0.66410
[29]	validation_0-error:0.40108	validation_0-logloss:0.66182	valida
tion_1-error:0.40332	validation_1-logloss:0.66409
[30]	validation_0-error:0.40111	validation_0-logloss:0.66174	valida
tion_1-error:0.40358	validation_1-logloss:0.66414
[31]	validation_0-error:0.40061	validation_0-logloss:0.66159	valida
tion_1-error:0.40497	validation_1-logloss:0.66417
[32]	validation_0-error:0.40018	validation_0-logloss:0.66138	valida
tion_1-error:0.40599	validation_1-logloss:0.66412
[33]	validation_0-error:0.39949	validation_0-logloss:0.66124	valida
tion_1-error:0.40535	validation_1-logloss:0.66429
[34]	validation_0-error:0.39895	validation_0-logloss:0.66105	valida
tion_1-error:0.40548	validation_1-logloss:0.66436
[35]	validation_0-error:0.39864	validation_0-logloss:0.66083	valida
tion_1-error:0.40535	validation_1-logloss:0.66420
[36]	validation_0-error:0.39795	validation_0-logloss:0.66069	valida
tion_1-error:0.40383	validation_1-logloss:0.66415
[37]	validation_0-error:0.39777	validation_0-logloss:0.66048	valida
tion_1-error:0.40485	validation_1-logloss:0.66395
[38]	validation_0-error:0.39783	validation_0-logloss:0.66036	valida
tion_1-error:0.40459	validation_1-logloss:0.66399
[39]	validation_0-error:0.39770	validation_0-logloss:0.66024	valida
tion_1-error:0.40459	validation_1-logloss:0.66405
[40]	validation_0-error:0.39783	validation_0-logloss:0.66018	valida
tion_1-error:0.40472	validation_1-logloss:0.66410
[41]	validation_0-error:0.39767	validation_0-logloss:0.66008	valida
tion_1-error:0.40459	validation_1-logloss:0.66415
[42]	validation_0-error:0.39726	validation_0-logloss:0.65994	valida
tion_1-error:0.40561	validation_1-logloss:0.66419
[43]	validation_0-error:0.39695	validation_0-logloss:0.65983	valida
tion_1-error:0.40586	validation_1-logloss:0.66428
[44]	validation_0-error:0.39670	validation_0-logloss:0.65967	valida
```

```
tion_1-error:0.40599	validation_1-logloss:0.66411
[45]	validation_0-error:0.39604	validation_0-logloss:0.65953	valida
tion_1-error:0.40345	validation_1-logloss:0.66408
[46]	validation_0-error:0.39517	validation_0-logloss:0.65943	valida
tion_1-error:0.40294	validation_1-logloss:0.66422
[47]	validation_0-error:0.39501	validation_0-logloss:0.65928	valida
tion_1-error:0.40294	validation_1-logloss:0.66395
[48]	validation_0-error:0.39476	validation_0-logloss:0.65916	valida
tion_1-error:0.40269	validation_1-logloss:0.66378
[49]	validation_0-error:0.39470	validation_0-logloss:0.65912	valida
tion_1-error:0.40307	validation_1-logloss:0.66382
[50]	validation_0-error:0.39451	validation_0-logloss:0.65906	valida
tion_1-error:0.40307	validation_1-logloss:0.66388
[51]	validation_0-error:0.39457	validation_0-logloss:0.65896	valida
tion_1-error:0.40383	validation_1-logloss:0.66386
[52]	validation_0-error:0.39498	validation_0-logloss:0.65883	valida
tion_1-error:0.40282	validation_1-logloss:0.66382
[53]	validation_0-error:0.39485	validation_0-logloss:0.65875	valida
tion_1-error:0.40282	validation_1-logloss:0.66388
[54]	validation_0-error:0.39467	validation_0-logloss:0.65864	valida
tion_1-error:0.40269	validation_1-logloss:0.66376
[55]	validation_0-error:0.39470	validation_0-logloss:0.65861	valida
tion_1-error:0.40269	validation_1-logloss:0.66376
[56]	validation_0-error:0.39410	validation_0-logloss:0.65850	valida
tion_1-error:0.40129	validation_1-logloss:0.66371
[57]	validation_0-error:0.39354	validation_0-logloss:0.65839	valida
tion_1-error:0.40307	validation_1-logloss:0.66368
[58]	validation_0-error:0.39363	validation_0-logloss:0.65834	valida
tion_1-error:0.40320	validation_1-logloss:0.66369
[59]	validation_0-error:0.39326	validation_0-logloss:0.65824	valida
tion_1-error:0.40307	validation_1-logloss:0.66376
[60]	validation_0-error:0.39273	validation_0-logloss:0.65809	valida
tion_1-error:0.40244	validation_1-logloss:0.66381
[61]	validation_0-error:0.39257	validation_0-logloss:0.65801	valida
tion_1-error:0.40231	validation_1-logloss:0.66391
[62]	validation_0-error:0.39248	validation_0-logloss:0.65790	valida
tion_1-error:0.40307	validation_1-logloss:0.66391
[63]	validation_0-error:0.39288	validation_0-logloss:0.65785	valida
tion_1-error:0.40320	validation_1-logloss:0.66398
[64]	validation_0-error:0.39257	validation_0-logloss:0.65781	valida
tion_1-error:0.40294	validation_1-logloss:0.66400
[65]	validation_0-error:0.39244	validation_0-logloss:0.65775	valida
tion_1-error:0.40294	validation_1-logloss:0.66403
[66]	validation_0-error:0.39213	validation_0-logloss:0.65755	valida
tion_1-error:0.40193	validation_1-logloss:0.66403
[67]	validation_0-error:0.39097	validation_0-logloss:0.65744	valida
tion_1-error:0.40206	validation_1-logloss:0.66375
[68]	validation_0-error:0.39126	validation_0-logloss:0.65733	valida
tion_1-error:0.40244	validation_1-logloss:0.66375
[69]	validation_0-error:0.39122	validation_0-logloss:0.65721	valida
tion_1-error:0.40231	validation_1-logloss:0.66361
[70]	validation_0-error:0.39169	validation_0-logloss:0.65712	valida
tion_1-error:0.40155	validation_1-logloss:0.66360
[71]	validation_0-error:0.39163	validation_0-logloss:0.65700	valida
tion_1-error:0.40129	validation_1-logloss:0.66362
[72]	validation_0-error:0.39113	validation_0-logloss:0.65691	valida
tion_1-error:0.40079	validation_1-logloss:0.66364
```

```
[73]	validation_0-error:0.39097	validation_0-logloss:0.65681	valida
tion_1-error:0.40066	validation_1-logloss:0.66365
[74]	validation_0-error:0.39100	validation_0-logloss:0.65671	valida
tion_1-error:0.40117	validation_1-logloss:0.66361
[75]	validation_0-error:0.39079	validation_0-logloss:0.65668	valida
tion_1-error:0.40066	validation_1-logloss:0.66360
[76]	validation_0-error:0.39044	validation_0-logloss:0.65656	valida
tion_1-error:0.40053	validation_1-logloss:0.66348
[77]	validation_0-error:0.39003	validation_0-logloss:0.65645	valida
tion_1-error:0.40117	validation_1-logloss:0.66335
[78]	validation_0-error:0.39022	validation_0-logloss:0.65638	valida
tion_1-error:0.40117	validation_1-logloss:0.66330
[79]	validation_0-error:0.39010	validation_0-logloss:0.65628	valida
tion_1-error:0.40091	validation_1-logloss:0.66331
[80]	validation_0-error:0.39003	validation_0-logloss:0.65624	valida
tion_1-error:0.40104	validation_1-logloss:0.66336
[81]	validation_0-error:0.38997	validation_0-logloss:0.65618	valida
tion_1-error:0.40091	validation_1-logloss:0.66341
[82]	validation_0-error:0.39010	validation_0-logloss:0.65614	valida
tion_1-error:0.40091	validation_1-logloss:0.66347
[83]	validation_0-error:0.39007	validation_0-logloss:0.65613	valida
tion_1-error:0.40079	validation_1-logloss:0.66345
[84]	validation_0-error:0.39000	validation_0-logloss:0.65609	valida
tion_1-error:0.40079	validation_1-logloss:0.66343
[85]	validation_0-error:0.39007	validation_0-logloss:0.65604	valida
tion_1-error:0.40079	validation_1-logloss:0.66347
[86]	validation_0-error:0.39000	validation_0-logloss:0.65597	valida
tion_1-error:0.40079	validation_1-logloss:0.66345
[87]	validation_0-error:0.39025	validation_0-logloss:0.65590	valida
tion_1-error:0.40066	validation_1-logloss:0.66350
[88]	validation_0-error:0.39035	validation_0-logloss:0.65589	valida
tion_1-error:0.40066	validation_1-logloss:0.66353
[89]	validation_0-error:0.39022	validation_0-logloss:0.65583	valida
tion_1-error:0.40091	validation_1-logloss:0.66352
[90]	validation_0-error:0.39013	validation_0-logloss:0.65574	valida
tion_1-error:0.40079	validation_1-logloss:0.66355
[91]	validation_0-error:0.39025	validation_0-logloss:0.65566	valida
tion_1-error:0.40053	validation_1-logloss:0.66357
[92]	validation_0-error:0.39063	validation_0-logloss:0.65554	valida
tion_1-error:0.40180	validation_1-logloss:0.66362
[93]	validation_0-error:0.39075	validation_0-logloss:0.65543	valida
tion_1-error:0.40129	validation_1-logloss:0.66358
[94]	validation_0-error:0.39025	validation_0-logloss:0.65535	valida
tion_1-error:0.40079	validation_1-logloss:0.66360
[95]	validation_0-error:0.38966	validation_0-logloss:0.65523	valida
tion_1-error:0.40104	validation_1-logloss:0.66335
[96]	validation_0-error:0.38966	validation_0-logloss:0.65508	valida
tion_1-error:0.40104	validation_1-logloss:0.66341
[97]	validation_0-error:0.38944	validation_0-logloss:0.65495	valida
tion_1-error:0.40079	validation_1-logloss:0.66349
[98]	validation_0-error:0.38931	validation_0-logloss:0.65483	valida
tion_1-error:0.40091	validation_1-logloss:0.66354
[99]	validation_0-error:0.38947	validation_0-logloss:0.65476	valida
tion_1-error:0.40231	validation_1-logloss:0.66343
0.5976909413854352
```

Out[26]:
```
<matplotlib.legend.Legend at 0x1588cfd00>
```



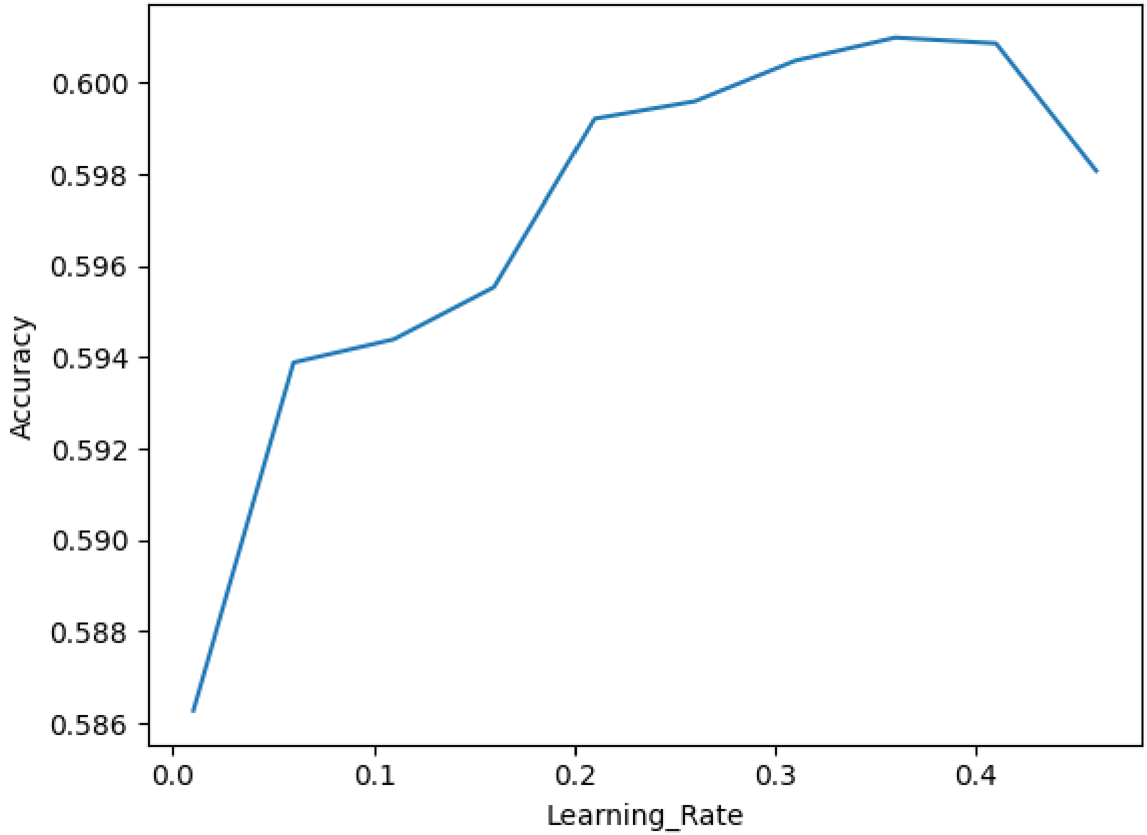
Training loss just started to converge so increase no of iterations or increase learning rate
Lets experiment with learning rate

Learning rate parameters vs accuracy

In [21]:
```
def plot_lr_vs_acc(train_X, train_Y, test_X, test_Y):
    acc_lis = []
    eta =[]
    for i in range(1,50,5):
        xgb_model = xgb.XGBClassifier(booster = 'dart', eta = i/100, max_depth
        xgb_model.fit(train_X, train_Y)
        y_pred = xgb_model.predict(test_X)
        acc = accuracy_score(test_Y, y_pred)
        eta.append(i/100)
        acc_lis.append(acc)

    plt.plot(eta, acc_lis)
    plt.xlabel("Learning_Rate")
    plt.ylabel("Accuracy")
    plt.show()

plot_lr_vs_acc(train_X, train_Y, test_X, test_Y)
```



Learning rate is peaking at 0.4 so trying that and checking

In [28]:
```
import xgboost as xgb
xgb_model = xgb.XGBClassifier(booster = 'dart', eta = 0.4, max_depth = 2, objec
eval_set = [(train_X, train_Y), (test_X, test_Y)]
xgb_model.fit(train_X, train_Y, eval_set=eval_set, eval_metric=["error", "loglo
y_pred = xgb_model.predict(test_X)
acc = accuracy_score(test_Y, y_pred)
print(acc)

import matplotlib.pyplot as plt
results = xgb_model.evals_result()
epochs = len(results['validation_0']['error'])
x_axis = range(0, epochs)
# plot log loss
fig, ax = plt.subplots()
ax.plot(x_axis, results['validation_0']['logloss'], label='Train')
ax.plot(x_axis, results['validation_1']['logloss'], label='Test')
ax.legend()
```

```
[0]	validation_0-error:0.41748	validation_0-logloss:0.68135	valida
tion_1-error:0.41804	validation_1-logloss:0.68053
[1]	validation_0-error:0.41304	validation_0-logloss:0.67593	valida
tion_1-error:0.41322	validation_1-logloss:0.67541
[2]	validation_0-error:0.41304	validation_0-logloss:0.67253	valida
tion_1-error:0.41335	validation_1-logloss:0.67165
[3]	validation_0-error:0.41219	validation_0-logloss:0.67099	valida
tion_1-error:0.41322	validation_1-logloss:0.67055
[4]	validation_0-error:0.41244	validation_0-logloss:0.66962	valida
tion_1-error:0.41271	validation_1-logloss:0.66920
[5]	validation_0-error:0.41241	validation_0-logloss:0.66825	valida
tion_1-error:0.41259	validation_1-logloss:0.66742
[6]	validation_0-error:0.40803	validation_0-logloss:0.66753	valida
tion_1-error:0.41018	validation_1-logloss:0.66718
[7]	validation_0-error:0.40809	validation_0-logloss:0.66680	valida
tion_1-error:0.41005	validation_1-logloss:0.66675
[8]	validation_0-error:0.40791	validation_0-logloss:0.66624	valida
tion_1-error:0.41005	validation_1-logloss:0.66633
[9]	validation_0-error:0.40797	validation_0-logloss:0.66569	valida
tion_1-error:0.41081	validation_1-logloss:0.66607
[10]	validation_0-error:0.40512	validation_0-logloss:0.66506	valida
tion_1-error:0.41030	validation_1-logloss:0.66592
[11]	validation_0-error:0.40478	validation_0-logloss:0.66474	valida
tion_1-error:0.40941	validation_1-logloss:0.66588
[12]	validation_0-error:0.40496	validation_0-logloss:0.66431	valida
tion_1-error:0.41005	validation_1-logloss:0.66576
[13]	validation_0-error:0.40424	validation_0-logloss:0.66387	valida
tion_1-error:0.40979	validation_1-logloss:0.66545
[14]	validation_0-error:0.40152	validation_0-logloss:0.66346	valida
tion_1-error:0.40586	validation_1-logloss:0.66519
[15]	validation_0-error:0.40190	validation_0-logloss:0.66302	valida
tion_1-error:0.40523	validation_1-logloss:0.66453
```

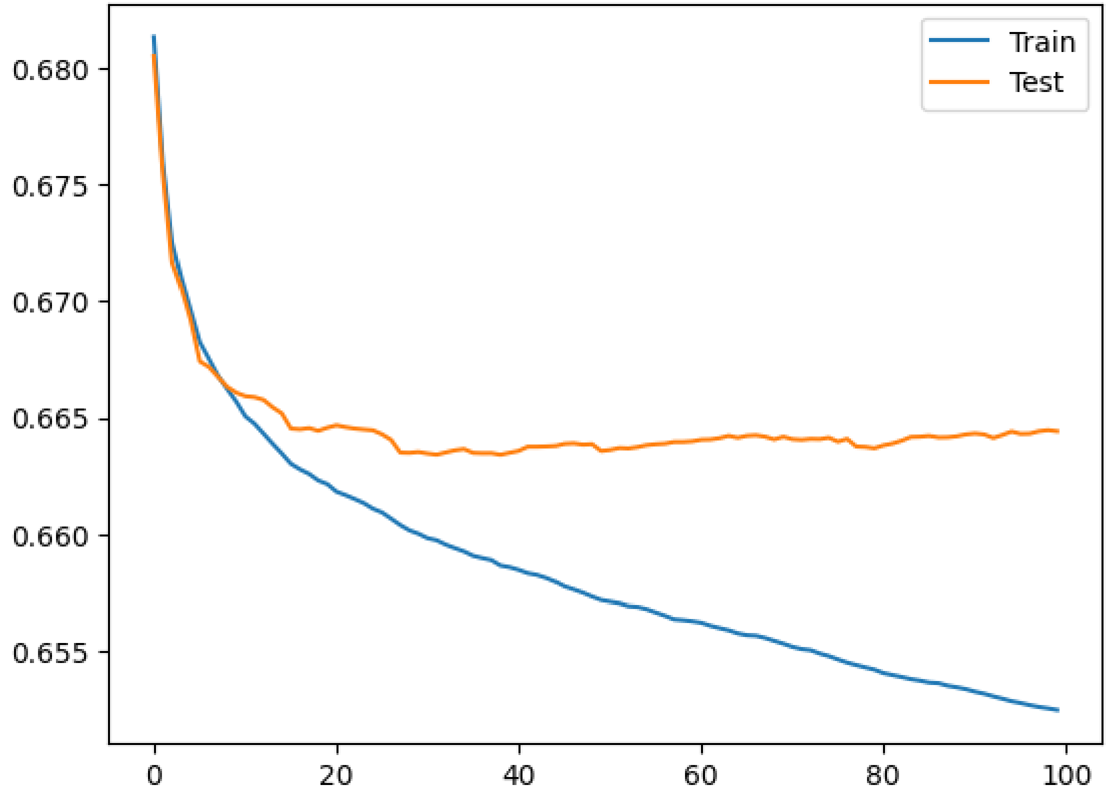
```
/Users/yamini/Library/Python/3.8/lib/python/site-packages/xgboost/sklearn.py:7
93: UserWarning: `eval_metric` in `fit` method is deprecated for better compat
ibility with scikit-learn, use `eval_metric` in constructor or`set_params` ins
tead.
  warnings.warn(
```

```
[16]	validation_0-error:0.40165	validation_0-logloss:0.66278	valida
tion_1-error:0.40345	validation_1-logloss:0.66450
[17]	validation_0-error:0.40115	validation_0-logloss:0.66258	valida
tion_1-error:0.40434	validation_1-logloss:0.66455
[18]	validation_0-error:0.40074	validation_0-logloss:0.66230	valida
tion_1-error:0.40586	validation_1-logloss:0.66443
[19]	validation_0-error:0.40039	validation_0-logloss:0.66214	valida
tion_1-error:0.40561	validation_1-logloss:0.66457
[20]	validation_0-error:0.40008	validation_0-logloss:0.66182	valida
tion_1-error:0.40510	validation_1-logloss:0.66467
[21]	validation_0-error:0.39986	validation_0-logloss:0.66167	valida
tion_1-error:0.40599	validation_1-logloss:0.66459
[22]	validation_0-error:0.39999	validation_0-logloss:0.66150	valida
tion_1-error:0.40459	validation_1-logloss:0.66453
[23]	validation_0-error:0.39980	validation_0-logloss:0.66133	valida
tion_1-error:0.40650	validation_1-logloss:0.66449
[24]	validation_0-error:0.39936	validation_0-logloss:0.66109	valida
tion_1-error:0.40738	validation_1-logloss:0.66446
[25]	validation_0-error:0.39955	validation_0-logloss:0.66093	valida
tion_1-error:0.40624	validation_1-logloss:0.66429
[26]	validation_0-error:0.39964	validation_0-logloss:0.66066	valida
tion_1-error:0.40738	validation_1-logloss:0.66405
[27]	validation_0-error:0.39939	validation_0-logloss:0.66039	valida
tion_1-error:0.40675	validation_1-logloss:0.66350
[28]	validation_0-error:0.39914	validation_0-logloss:0.66016	valida
tion_1-error:0.40802	validation_1-logloss:0.66349
[29]	validation_0-error:0.39946	validation_0-logloss:0.66001	valida
tion_1-error:0.40738	validation_1-logloss:0.66352
[30]	validation_0-error:0.39911	validation_0-logloss:0.65982	valida
tion_1-error:0.40738	validation_1-logloss:0.66346
[31]	validation_0-error:0.39895	validation_0-logloss:0.65973	valida
tion_1-error:0.40726	validation_1-logloss:0.66341
[32]	validation_0-error:0.39886	validation_0-logloss:0.65954	valida
tion_1-error:0.40713	validation_1-logloss:0.66350
[33]	validation_0-error:0.39855	validation_0-logloss:0.65939	valida
tion_1-error:0.40713	validation_1-logloss:0.66360
[34]	validation_0-error:0.39849	validation_0-logloss:0.65925	valida
tion_1-error:0.40650	validation_1-logloss:0.66365
[35]	validation_0-error:0.39780	validation_0-logloss:0.65906	valida
tion_1-error:0.40573	validation_1-logloss:0.66349
[36]	validation_0-error:0.39758	validation_0-logloss:0.65896	valida
tion_1-error:0.40586	validation_1-logloss:0.66347
[37]	validation_0-error:0.39786	validation_0-logloss:0.65888	valida
tion_1-error:0.40586	validation_1-logloss:0.66347
[38]	validation_0-error:0.39795	validation_0-logloss:0.65864	valida
tion_1-error:0.40497	validation_1-logloss:0.66340
[39]	validation_0-error:0.39802	validation_0-logloss:0.65857	valida
tion_1-error:0.40472	validation_1-logloss:0.66349
[40]	validation_0-error:0.39792	validation_0-logloss:0.65846	valida
tion_1-error:0.40447	validation_1-logloss:0.66357
[41]	validation_0-error:0.39758	validation_0-logloss:0.65832	valida
tion_1-error:0.40472	validation_1-logloss:0.66375
[42]	validation_0-error:0.39739	validation_0-logloss:0.65824	valida
tion_1-error:0.40485	validation_1-logloss:0.66375
[43]	validation_0-error:0.39708	validation_0-logloss:0.65811	valida
tion_1-error:0.40447	validation_1-logloss:0.66376
[44]	validation_0-error:0.39701	validation_0-logloss:0.65795	valida
```

```
tion_1-error:0.40421	validation_1-logloss:0.66378
[45]	validation_0-error:0.39698	validation_0-logloss:0.65775	valida
tion_1-error:0.40523	validation_1-logloss:0.66388
[46]	validation_0-error:0.39692	validation_0-logloss:0.65762	valida
tion_1-error:0.40459	validation_1-logloss:0.66390
[47]	validation_0-error:0.39676	validation_0-logloss:0.65748	valida
tion_1-error:0.40472	validation_1-logloss:0.66384
[48]	validation_0-error:0.39592	validation_0-logloss:0.65732	valida
tion_1-error:0.40472	validation_1-logloss:0.66386
[49]	validation_0-error:0.39589	validation_0-logloss:0.65717	valida
tion_1-error:0.40358	validation_1-logloss:0.66357
[50]	validation_0-error:0.39579	validation_0-logloss:0.65710	valida
tion_1-error:0.40358	validation_1-logloss:0.66361
[51]	validation_0-error:0.39385	validation_0-logloss:0.65703	valida
tion_1-error:0.40332	validation_1-logloss:0.66370
[52]	validation_0-error:0.39388	validation_0-logloss:0.65691	valida
tion_1-error:0.40409	validation_1-logloss:0.66368
[53]	validation_0-error:0.39385	validation_0-logloss:0.65687	valida
tion_1-error:0.40409	validation_1-logloss:0.66374
[54]	validation_0-error:0.39370	validation_0-logloss:0.65678	valida
tion_1-error:0.40409	validation_1-logloss:0.66382
[55]	validation_0-error:0.39363	validation_0-logloss:0.65664	valida
tion_1-error:0.40535	validation_1-logloss:0.66386
[56]	validation_0-error:0.39360	validation_0-logloss:0.65650	valida
tion_1-error:0.40459	validation_1-logloss:0.66388
[57]	validation_0-error:0.39363	validation_0-logloss:0.65634	valida
tion_1-error:0.40282	validation_1-logloss:0.66396
[58]	validation_0-error:0.39351	validation_0-logloss:0.65631	valida
tion_1-error:0.40307	validation_1-logloss:0.66396
[59]	validation_0-error:0.39323	validation_0-logloss:0.65626	valida
tion_1-error:0.40294	validation_1-logloss:0.66398
[60]	validation_0-error:0.39320	validation_0-logloss:0.65620	valida
tion_1-error:0.40282	validation_1-logloss:0.66405
[61]	validation_0-error:0.39270	validation_0-logloss:0.65607	valida
tion_1-error:0.40282	validation_1-logloss:0.66406
[62]	validation_0-error:0.39263	validation_0-logloss:0.65597	valida
tion_1-error:0.40307	validation_1-logloss:0.66412
[63]	validation_0-error:0.39235	validation_0-logloss:0.65588	valida
tion_1-error:0.40269	validation_1-logloss:0.66422
[64]	validation_0-error:0.39216	validation_0-logloss:0.65575	valida
tion_1-error:0.40231	validation_1-logloss:0.66414
[65]	validation_0-error:0.39210	validation_0-logloss:0.65567	valida
tion_1-error:0.40218	validation_1-logloss:0.66422
[66]	validation_0-error:0.39198	validation_0-logloss:0.65564	valida
tion_1-error:0.40244	validation_1-logloss:0.66424
[67]	validation_0-error:0.39191	validation_0-logloss:0.65555	valida
tion_1-error:0.40206	validation_1-logloss:0.66418
[68]	validation_0-error:0.39185	validation_0-logloss:0.65542	valida
tion_1-error:0.40091	validation_1-logloss:0.66406
[69]	validation_0-error:0.39110	validation_0-logloss:0.65530	valida
tion_1-error:0.40129	validation_1-logloss:0.66418
[70]	validation_0-error:0.39113	validation_0-logloss:0.65517	valida
tion_1-error:0.40117	validation_1-logloss:0.66407
[71]	validation_0-error:0.39057	validation_0-logloss:0.65507	valida
tion_1-error:0.40091	validation_1-logloss:0.66405
[72]	validation_0-error:0.39057	validation_0-logloss:0.65502	valida
tion_1-error:0.40079	validation_1-logloss:0.66409
```

```
[73]	validation_0-error:0.39025	validation_0-logloss:0.65488	valida
tion_1-error:0.39990	validation_1-logloss:0.66408
[74]	validation_0-error:0.38972	validation_0-logloss:0.65477	valida
tion_1-error:0.40015	validation_1-logloss:0.66414
[75]	validation_0-error:0.38991	validation_0-logloss:0.65463	valida
tion_1-error:0.39964	validation_1-logloss:0.66397
[76]	validation_0-error:0.38891	validation_0-logloss:0.65449	valida
tion_1-error:0.39952	validation_1-logloss:0.66409
[77]	validation_0-error:0.38872	validation_0-logloss:0.65438	valida
tion_1-error:0.40066	validation_1-logloss:0.66376
[78]	validation_0-error:0.38894	validation_0-logloss:0.65428	valida
tion_1-error:0.40142	validation_1-logloss:0.66374
[79]	validation_0-error:0.38863	validation_0-logloss:0.65419	valida
tion_1-error:0.40091	validation_1-logloss:0.66369
[80]	validation_0-error:0.38800	validation_0-logloss:0.65404	valida
tion_1-error:0.40041	validation_1-logloss:0.66381
[81]	validation_0-error:0.38725	validation_0-logloss:0.65395	valida
tion_1-error:0.39964	validation_1-logloss:0.66389
[82]	validation_0-error:0.38712	validation_0-logloss:0.65387	valida
tion_1-error:0.39990	validation_1-logloss:0.66401
[83]	validation_0-error:0.38694	validation_0-logloss:0.65378	valida
tion_1-error:0.39977	validation_1-logloss:0.66417
[84]	validation_0-error:0.38697	validation_0-logloss:0.65372	valida
tion_1-error:0.39977	validation_1-logloss:0.66418
[85]	validation_0-error:0.38722	validation_0-logloss:0.65363	valida
tion_1-error:0.39990	validation_1-logloss:0.66422
[86]	validation_0-error:0.38716	validation_0-logloss:0.65360	valida
tion_1-error:0.39990	validation_1-logloss:0.66415
[87]	validation_0-error:0.38703	validation_0-logloss:0.65349	valida
tion_1-error:0.40003	validation_1-logloss:0.66416
[88]	validation_0-error:0.38706	validation_0-logloss:0.65343	valida
tion_1-error:0.40003	validation_1-logloss:0.66420
[89]	validation_0-error:0.38687	validation_0-logloss:0.65335	valida
tion_1-error:0.40015	validation_1-logloss:0.66428
[90]	validation_0-error:0.38716	validation_0-logloss:0.65324	valida
tion_1-error:0.39964	validation_1-logloss:0.66432
[91]	validation_0-error:0.38728	validation_0-logloss:0.65315	valida
tion_1-error:0.39977	validation_1-logloss:0.66428
[92]	validation_0-error:0.38709	validation_0-logloss:0.65304	valida
tion_1-error:0.39914	validation_1-logloss:0.66413
[93]	validation_0-error:0.38647	validation_0-logloss:0.65294	valida
tion_1-error:0.40053	validation_1-logloss:0.66426
[94]	validation_0-error:0.38628	validation_0-logloss:0.65283	valida
tion_1-error:0.40066	validation_1-logloss:0.66441
[95]	validation_0-error:0.38597	validation_0-logloss:0.65275	valida
tion_1-error:0.39977	validation_1-logloss:0.66430
[96]	validation_0-error:0.38547	validation_0-logloss:0.65266	valida
tion_1-error:0.39926	validation_1-logloss:0.66431
[97]	validation_0-error:0.38547	validation_0-logloss:0.65258	valida
tion_1-error:0.39914	validation_1-logloss:0.66442
[98]	validation_0-error:0.38540	validation_0-logloss:0.65252	valida
tion_1-error:0.39926	validation_1-logloss:0.66446
[99]	validation_0-error:0.38540	validation_0-logloss:0.65245	valida
tion_1-error:0.39926	validation_1-logloss:0.66442
0.600735853844202
```

Out[28]: `<matplotlib.legend.Legend at 0x1589eb940>`



Learning rate 0.3 is 0.597 whereas learning rate 0.4 is 0.600 - there is a slight difference and this tuning is as well on the test set So there might be a good chance that since this model is specifically tuned on test data so any other data which is slightly different from test data might predict with a lower accuracy

In [ ]:

Method 2 : GridSearchCV

In [23]:

```
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV
estimator = XGBClassifier(
    objective= 'binary:logistic',
    nthread=4,
    seed=42
)
parameters = {
    'max_depth': range (2, 10, 1),
    'n_estimators': range(60, 220, 40),
    'learning_rate': [0.1, 0.01, 0.05]
}
grid_search = GridSearchCV(
    estimator=estimator,
    param_grid=parameters,
    scoring = 'roc_auc',
    n_jobs = 10,
    cv = 10,
    verbose=True
)
grid_search.fit(train_X, train_Y)
```

```
Fitting 10 folds for each of 96 candidates, totalling 960 fits
```

Out[23]:

```
▸ GridSearchCV
▸ estimator: XGBClassifier
  ▸ XGBClassifier
```

In [24]:
```
grid_search.score(test_X, test_Y)
```

Out[24]: `0.6316219155688723`

In [29]:
```
grid_search.best_params_
```

Out[29]: `{'learning_rate': 0.05, 'max_depth': 2, 'n_estimators': 140}`

Best performing model is 63% at learning rate 0.05 and n_estimator = 140

In [ ]:
```

```

In [ ]:
```

```

2. Now compare the XGBoost performance to the decision tree implementation from part
   3. Describe in text how they compare, and if this aligns with what you expect. (3 points)

Place your code here

```
Decision tree was over fitting for max_depth of None, 128, 256
and 512 so I took max_depth of 8 and the test set was
performing around 59% whereas xgboost is around 60% with method
1 and 63% in method 2


XGBoost Features
Regularized Learning: Regularization term helps to smooth the
final learnt weights to avoid over-fitting. The regularized
objective will tend to select a model employing simple and
predictive functions.
Gradient Tree Boosting: The tree ensemble model cannot be
optimized using traditional optimization methods in Euclidean
space. Instead, the model is trained in an additive manner.
Shrinkage and Column Subsampling: Besides the regularized
objective, two additional techniques are used to further
prevent overfitting. The first technique is shrinkage
introduced by Friedman. Shrinkage scales newly added weights by
a factor η after each step of tree boosting. Similar to a
learning rate in stochastic optimization, shrinkage reduces the
influence of each tree and leaves space for future trees to
improve the model.
XGBoost is an ensemble learning method
```

```
In the case of boosting, the decision tree followed a
sequential chain for learning. Each split sub-parts gets
trained from its forerunner, and any kind of error existing in
the current part gets rectified and leads to the next sub-part.
The above description clarifies that in the case of boosting
techniques, the initial stage base learner holds a weaker
nature and continues to generate stronger variants of learners
as the tree expands. Each of the strong learners provides
crucial data for final prediction. Sometimes, to generate more
strong learner variants, several weak and stronger learners are
fused.
-Regularisation
-Weighted quantile sketch
-Block structure for parallel learning


Comparison:

XgBoost was bound to do better as it builds multiple trees -
Each new tree is built to improve on the deficiencies of the
previous trees and this concept is called boosting.
when compared to
Decision tree that has only one tree so its deficiency lies
there with no other chance for it to change itself


XgBoost also uses gradient of the previous tree into the new
tree which helps in learning and retaining previous tree
information
unlike Decision tree that has only one tree which gives less
flexibility to learn
```

End of homework