1) A)

Output of the given code (Generated Table)

DecisionTreeClassifier

Dataset	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
australian	72.61%	74.63%	75.52%	77.53%	77.97%	79.86%	83.05%	81.29%	80.14%	82.91%
balance-scale	70.10%	72.47%	71.20%	75.69%	73.77%	75.67%	77.74%	75.99%	78.09%	76.98%
hypothyroid	94.94%	96.31%	97.77%	99.18%	99.21%	99.42%	99.42%	99.52%	99.34%	99.20%

BernoulliNB with priors

Dataset	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
australian	73.47%	79.85%	81.72%	80.43%	79.69%	79.84%	80.12%	81.14%	82.16%	81.28%
balance-scale	46.08%	46.08%	46.08%	46.08%	46.08%	46.08%	46.08%	46.08%	46.08%	46.08%
hypothyroid	91.38%	91.81%	92.23%	92.23%	92.23%	92.26%	92.23%	92.23%	92.23%	92.23%

B) 3 and 5

C) (1) BNB performs better with priors as default (class_prior = None).

2) When Random State is left default:-

A)

Model accuracy (Testing Set) = 0.842966292134831

Model accuracy (Training Set) = 0. 8564516129032258

ROC_AUC SCORE (Testing Set) = 0.8215654395540587

ROC_AUC SCORE (Training Set) = 0.829507269521243

B)

HYPERPARAMETER SEARCH TUNING WITH TRAINING SET :

Optimal score = 0.8466501506510756

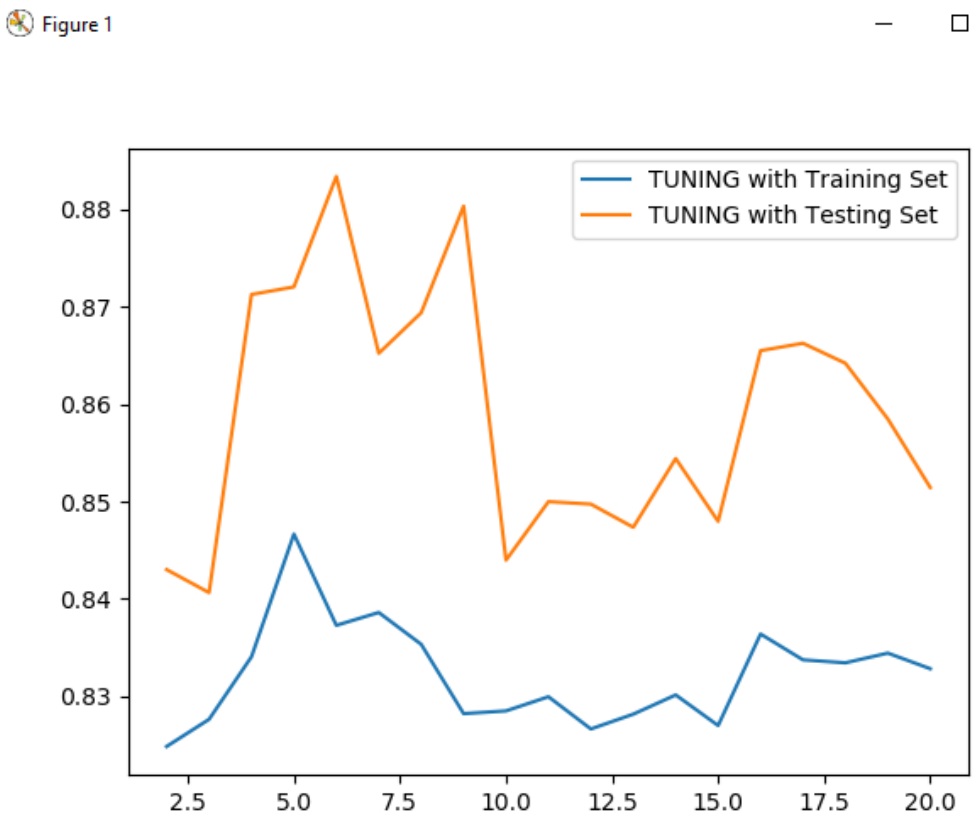
Grid Best Params = {'min_samples_leaf': 5}

HYPERPARAMETER SEARCH TUNING WITH TESTING SET :

Optimal score = 0.8833682446585673

Grid Best Params = {'min_samples_leaf': 6}

C)



D)

PROBABILITY OF A WOMAN FIRST CLASS PASSENGER TO SURVIVE IS 38.57338017174081

3) When Random_State = 0:-

A)

Model accuracy (Testing Set) = 0.8277153558052435
Model accuracy (Training Set) = 0.864516129032258
ROC_AUC SCORE (Testing Set) = 0.8077601410934745
ROC_AUC SCORE (Training Set) = 0.8380650207665443

B)

HYPERPARAMETER SEARCH TUNING WITH TRAINING SET :

Optimal score = 0.8554181028005216

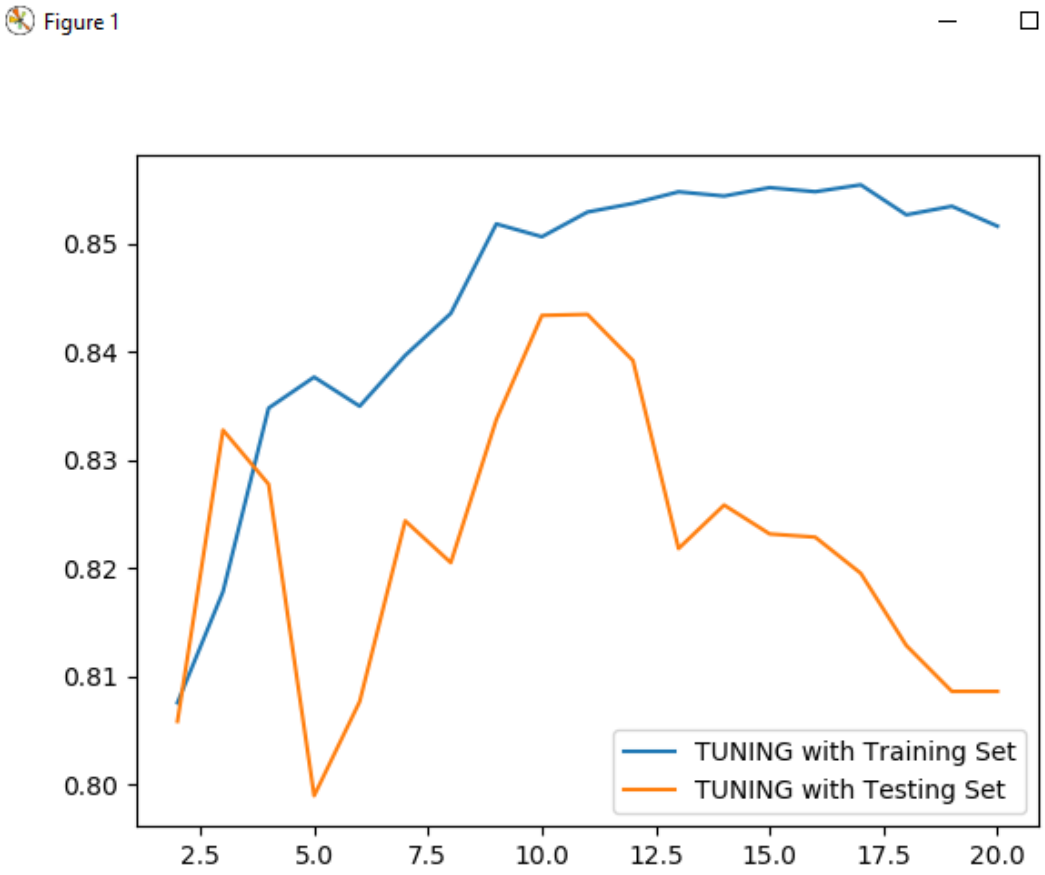
Grid Best Params = {'min_samples_leaf': 17}

HYPERPARAMETER SEARCH TUNING WITH TESTING SET :

Optimal score = 0.8434388528138527

Grid Best Params = {'min_samples_leaf': 11}

C)



D)

PROBABILITY OF A WOMAN FIRST CLASS PASSENGER TO SURVIVE IS 33.389355742296914

CODE:-

```
from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc, accuracy_score, roc_auc_score
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV

"""READING DATASET AND CREATING MODEL"""
df = pd.read_csv("titanic.csv")
inputs = df.drop("Survived",axis='columns')
target = df.Survived
normalised_inputs = (inputs - inputs.min())/(inputs.max() - inputs.min())
x_train,x_test,y_train,y_test =
train_test_split(inputs,target,test_size=0.3,random_state=0)
treeModel = tree.DecisionTreeClassifier(random_state=0)
treeModel.fit(x_train,y_train)

#MODEL ACCURACY FOR TESTING AND TRAINING SETS
print("Model accuracy (Testing Set) = ",treeModel.score(x_test,y_test))
print("Model accuracy (Training Set) = ",treeModel.score(x_train,y_train))

#ROC_AUC SCORE FOR TESTING AND TRAINING SETS
tree_roc_auc_test = roc_auc_score(y_test,treeModel.predict(x_test))
tree_roc_auc_train = roc_auc_score(y_train,treeModel.predict(x_train))
print("ROC_AUC SCORE (Testing Set) = ",tree_roc_auc_test)
print("ROC_AUC SCORE (Training Set) = ",tree_roc_auc_train)


#GRID SEARCH ALGO FOR TUNING HYPERPARAMETERS AND SEARCHING OPTIMAL MIN_SAMPLES_LEAF IN
TESTING AND TRAINING SETS
param_grid = dict(min_samples_leaf=range(2,21))

gridTraining = GridSearchCV(treeModel,param_grid,scoring = 'roc_auc')
gridTraining.fit(x_train,y_train)

gridTesting = GridSearchCV(treeModel,param_grid,scoring = 'roc_auc')
gridTesting.fit(x_test,y_test)

print("HYPERPARAMETER SEARCH TUNING WITH TRAINING SET :")
print("Optimal score = ",gridTraining.best_score_)
print("Grid Best Params = ",gridTraining.best_params_)

print("HYPERPARAMETER SEARCH TUNING WITH TESTING SET :")
print("Optimal score = ",gridTesting.best_score_)
print("Grid Best Params = ",gridTesting.best_params_)

plt.plot(range(2,21),gridTraining.cv_results_['mean_test_score'],label='TUNING with
Training Set')
plt.plot(range(2,21),gridTesting.cv_results_['mean_test_score'],label='TUNING with Testing
Set')
plt.legend()
plt.show()


""" PREDICTING POSTERIOR PROBABILITY """
inputSet_for_prediction = []
for i in range(len(inputs['Pclass'])):
    if inputs['Pclass'][i]==1 and inputs['Sex'][i] == 1:

inputSet_for_prediction.append([inputs['Pclass'][i],inputs['Sex'][i],inputs['Age'][i],inpu
ts['Siblings_Spouses_Aboard'][i],inputs['Parents_Children_Aboard'][i]])


probs = treeModel.predict_proba(inputSet_for_prediction)[:,1]

print("PROBABILITY OF A WOMAN FIRST CLASS PASSENGER TO SURVIVE IS
",(sum(probs)/len(probs))*100)
```