Question 1a

- Answer

Misclassification rate of the classification tree on the training data at iteration 0 is 0.16736309654717396.

- Explanation

I built a classification tree on the training partition specified in the problem. Then I used the predcit_proba() function to get the misclassification rate for iter 0.

Question 1b

- Answer

Misclassification rate of the classification tree on the training data at iteration 1 is 0.153702635410365.

- Explanation

Using Professor Lam's code, I iterated to 0 and printed the misclassification rate.

Question 1c

- Answer

The Misclassification Rate of the classification tree on the Training data when the iteration converges is $8.744448964925766 \times 10^{-8}$.

- Code

```
# Build a classification tree on the training partition
w_train = np.full(nObs, 1.0)
accuracy = np.zeros(50)
misclassification = np.zeros(50)
ensemblePredProb = np.zeros((nObs, 2))
for iter in range(50):
    classTree = tree.DecisionTreeClassifier(criterion='entropy', max_depth=5, random_state=20210415)
    treeFit = classTree.fit(x_train, y_train, w_train)
    treePredProb = classTree.predict_proba(x_train)
    accuracy[iter] = classTree.score(x_train, y_train, w_train)
    misclassification[iter] = 1 - accuracy[iter]
    ensemblePredProb += accuracy[iter] * treePredProb
    if (accuracy[iter] >= 0.9999999):
        break
    # Update the weights
    eventError = np.where(y_train == 1, (1 - treePredProb[:,1]), (0 - treePredProb[:,1]))
    predClass = np.where(treePredProb[:,1] >= 0.2, 1, 0)
    w_train = np.where(predClass != y_train, 2 + np.abs(eventError), np.abs(eventError))
    if iter == 0 or iter == 1:
        print('at iter', iter, ', Misclassification Rate = ', misclassification[iter])
```

- Explanation

Using Professor Lam's code, I iterated from 0 to 50 and calculated the misclassification rate for each value of iter by doing 1 - accuracy rate, which is calculated by using the classTree library with the predict_proba() function.

Question 1d

- Answer

The Area Under Curve metric on the Testing data using the final converged classification tree is 0.3222911912368781.

- Code

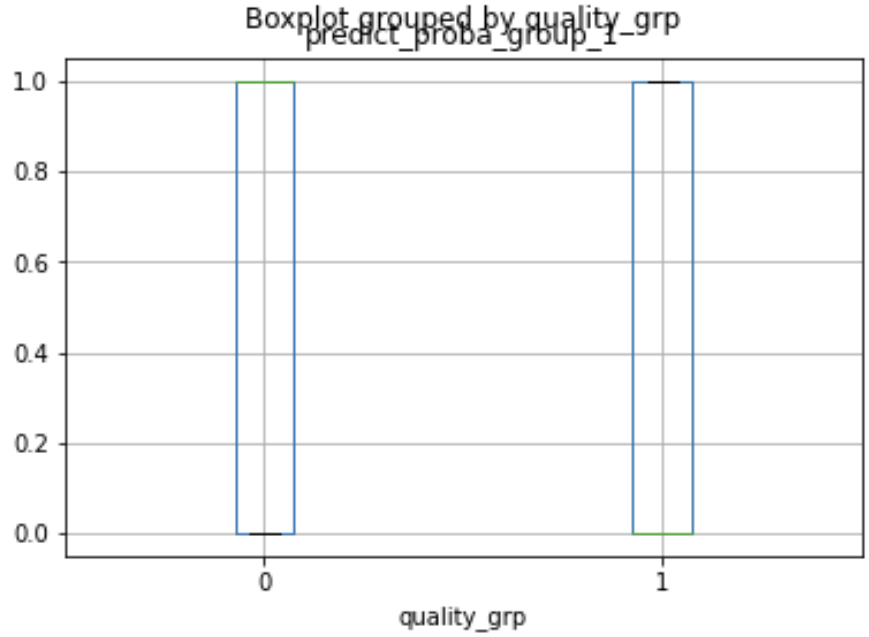
```
y_score = treeFit.predict_proba(x_test)
AUC = metrics.roc_auc_score(y_test, y_score[:,1])
print('AUC = ', AUC)
```

- Explanation

I used the predict_proba() function from the treeFit library and the roc_auc_score() function from the metrics library to calculate the area under the curve.

Question 1e

- Boxplot



- Code

```
bagPredProb = treeFit.predict_proba(x_test)
testData['predict_proba_group_0'] = bagPredProb[:, 0]
testData['predict_proba_group_1'] = bagPredProb[:, 1]
testData.boxplot(column='predict_proba_group_1', by='quality_grp')
plt.show()
```

- Explanation

I added two columns to the testData data frame: one with predicted_proba for quality_grp 0 and one for quality_grp 1. Then I created a boxplot based on the predicted probabilities of column 1. The predict_proba() function returns an array having all the categories, like [0, 1]. I'm assuming that the question is asking to pick only category 1 from the predict_proba() function.

Question 2a

- Answer

Model 5 = Intercept + alcohol + free_sulfur_dioxide + sulphates + citric_acid + residual_sugar

- Code

```
# Model 5
fifth_model = [('alcohol','free_sulfur_dioxide','sulphates','citric_acid','residual_sugar')]
for r in range(1):
    modelTerm = list(fifth_model[r])
    trainData2 = trainData[modelTerm].dropna()
    trainData2 = stats.add_constant(trainData, prepend = True)
    LLK1, DF1, fullParams1, thisFit = build_mnlogit(trainData, y)
    testDev = 2.0 * (LLK1 - LLK0)
    testDF = DF1 - DF0
    testPValue = scipy.stats.chi2.sf(testDev, testDF)
    if testPValue < 0.05:
        chi_dict[fifth_model[r]] = testPValue
key_min = min(chi_dict.keys(), key=(lambda k: chi_dict[k]))
print('Model 5 = Intercept +',' + '.join(''.join(t) for t in key_min))
```

Question 2b

- Answer

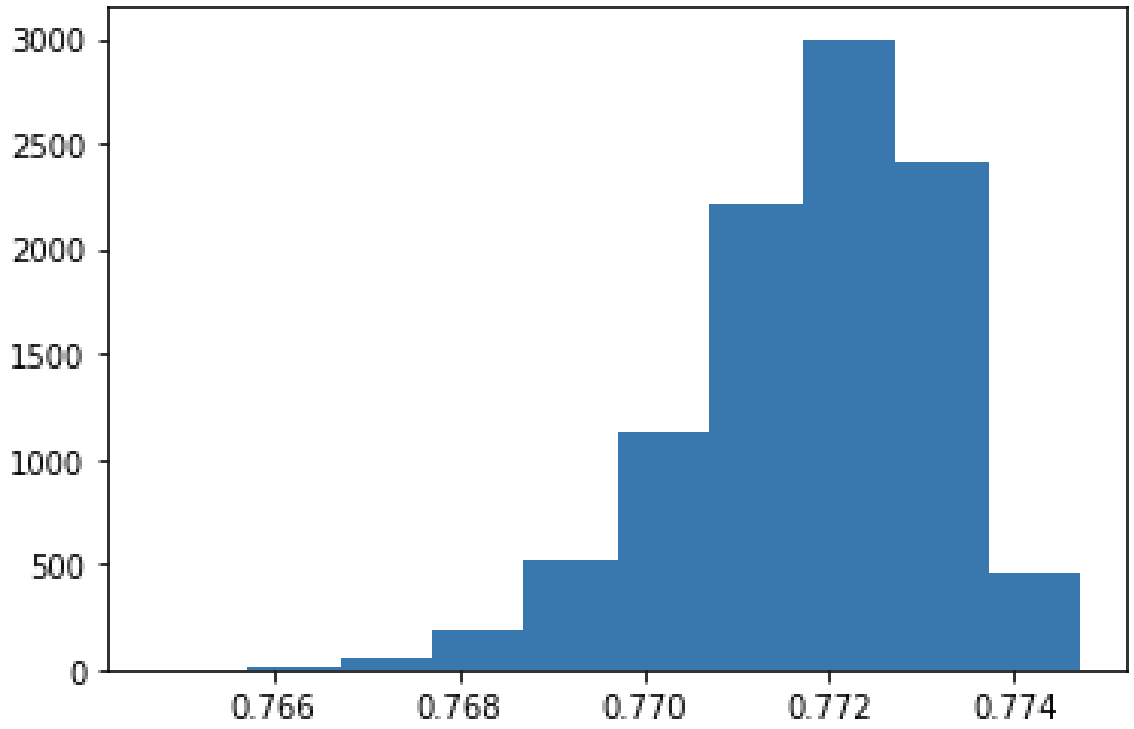
AUC: 0.7723613128085971

- Code/Explanation

```
clf = LogisticRegression(random_state=0).fit(x_train, y_train)
y_score = clf.predict_proba(x_test)
auc = metrics.roc_auc_score(y_test, y_score[:,1])
print("AUC: ", auc)
```

Question 2c

- Answer



- Code/Explanation

```
train_sample = pd.concat([x_train,y_train],axis=1)
train_sample = train_sample.values

random.seed(20210415)
def sample_wr(inData):
    n = len(inData)
    outData = np.empty((n,6))
    for i in range(n):
        j = int(random.random() * n)
        outData[i] = inData[j]
    return outData

AUC_array = np.zeros(10000)
for i in range(10000):
    bootstrap = sample_wr(train_sample)
    x_train = bootstrap[:,:5]
    y_train = bootstrap[:,-1]
    logistic = LogisticRegression(random_state = 20210415).fit(x_train,y_train)
    pred_prob = logistic.predict_proba(x_test)
    AUC_array[i] = metrics.roc_auc_score(y_test, pred_prob[:,1])

plt.hist(AUC_array, bins=np.arange(min(AUC_array), max(AUC_array)+0.001, 0.001), align='mid')
plt.show()
```

Question 2d

- Answer

95% Confidence Interval: 0.7687049,0.7738949

- Code/Explanation

```
print('95% Confidence Interval: {:.7f},{:.7f}'
      .format(np.percentile(AUC_array, (2.5)), np.percentile(AUC_array, (97.5))))
```

Question 2e

- Answer

It is statistically significant from 0.5 because the problem says that if 0.50 falls within the confidence levels, then ‘statisticians will conclude that the AUC on the Testing data is not significantly different from 0.5’. From part (d), I got that the 95% Confidence interval is [0.7687049,0.7738949], meaning 0.50 does not fall in it, so it is statistically significant.