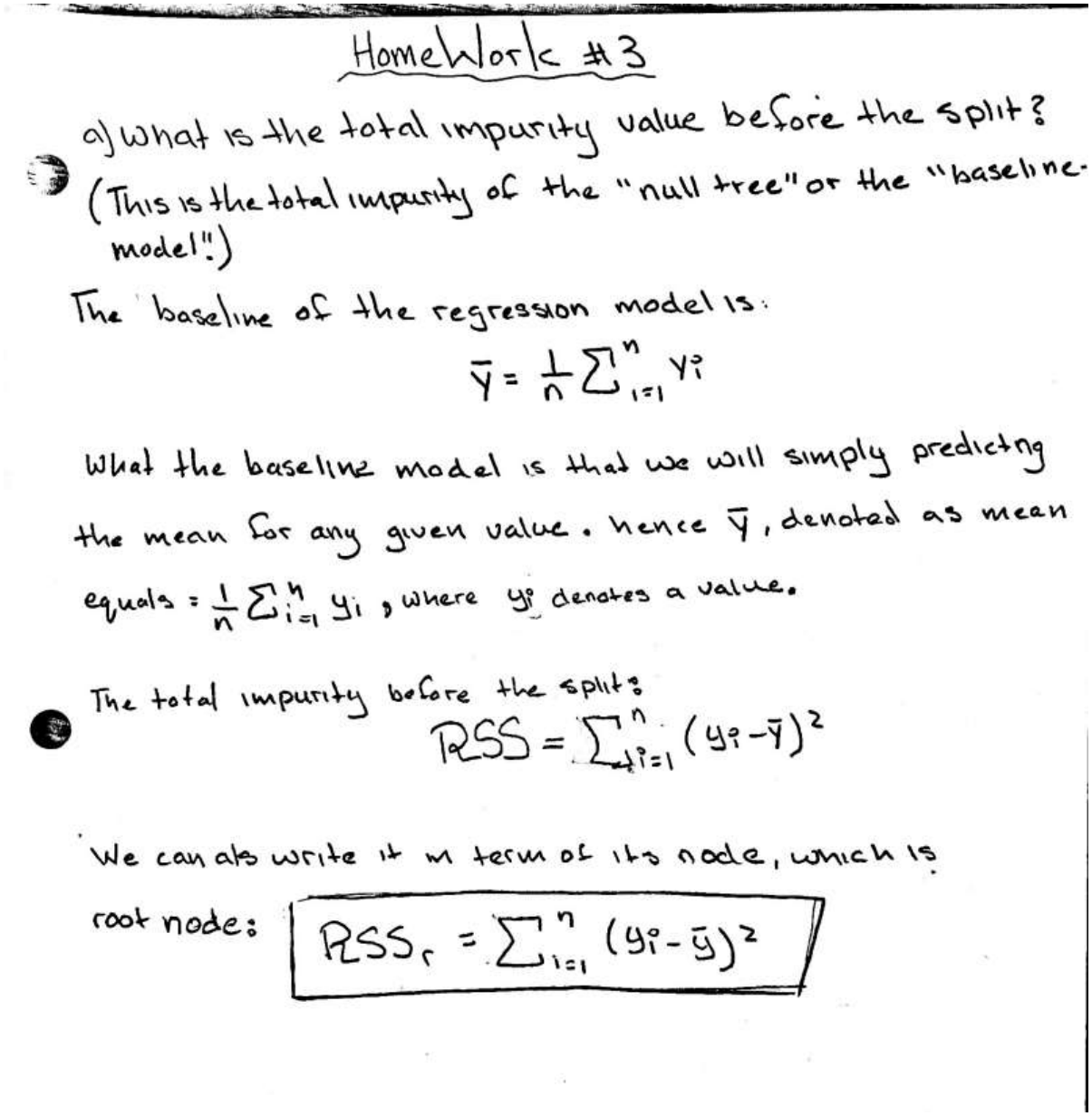
In [2]:
```
import pandas as pd
import statsmodels.formula.api as smf
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np
import statsmodels.api as sm
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import confusion_matrix
```

Problem 1: (40 points)

In this problem, we consider splitting when building a regression tree in the CART algorithm. We assume that there is a feature vector X ∈ R p and dependent variable Y ∈ R. We have collected a training dataset (x1, y1), . . . ,(xn, yn), where xi ∈ R p and yi ∈ R for all i = 1, . . . , n. We also assume, for simplicity, that we are considering the initial split at the top (root node) of the tree. An arbitrary split simply divides the training dataset into a partition of size two. By appropriately reshuffling the data, we can represent this partition (again for simplicity) via two sub-datasets (x1, y1), . . . ,(xN , yN) and (xN+1, yN+1), . . . ,(xn, yn) where N is the index of the last observation included in the first set. Assume throughout that our impurity function is the RSS error – the standard choice for a regression tree.

Part 1.A

a) (5 points) What is the total impurity value before the split? (This is the total impurity of the “null tree” or the “baseline model”.)



Homework #3

a) What is the total impurity value before the split? (This is the total impurity of the "null tree" or the "baseline-model".)

The baseline of the regression model is:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

What the baseline model is that we will simply predicting the mean for any given value. hence $\bar{y}$, denotad as mean equals $= \frac{1}{n}\sum_{i=1}^{n} y_i$, where y_i denotes a value.

The total impurity before the split:

$$RSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

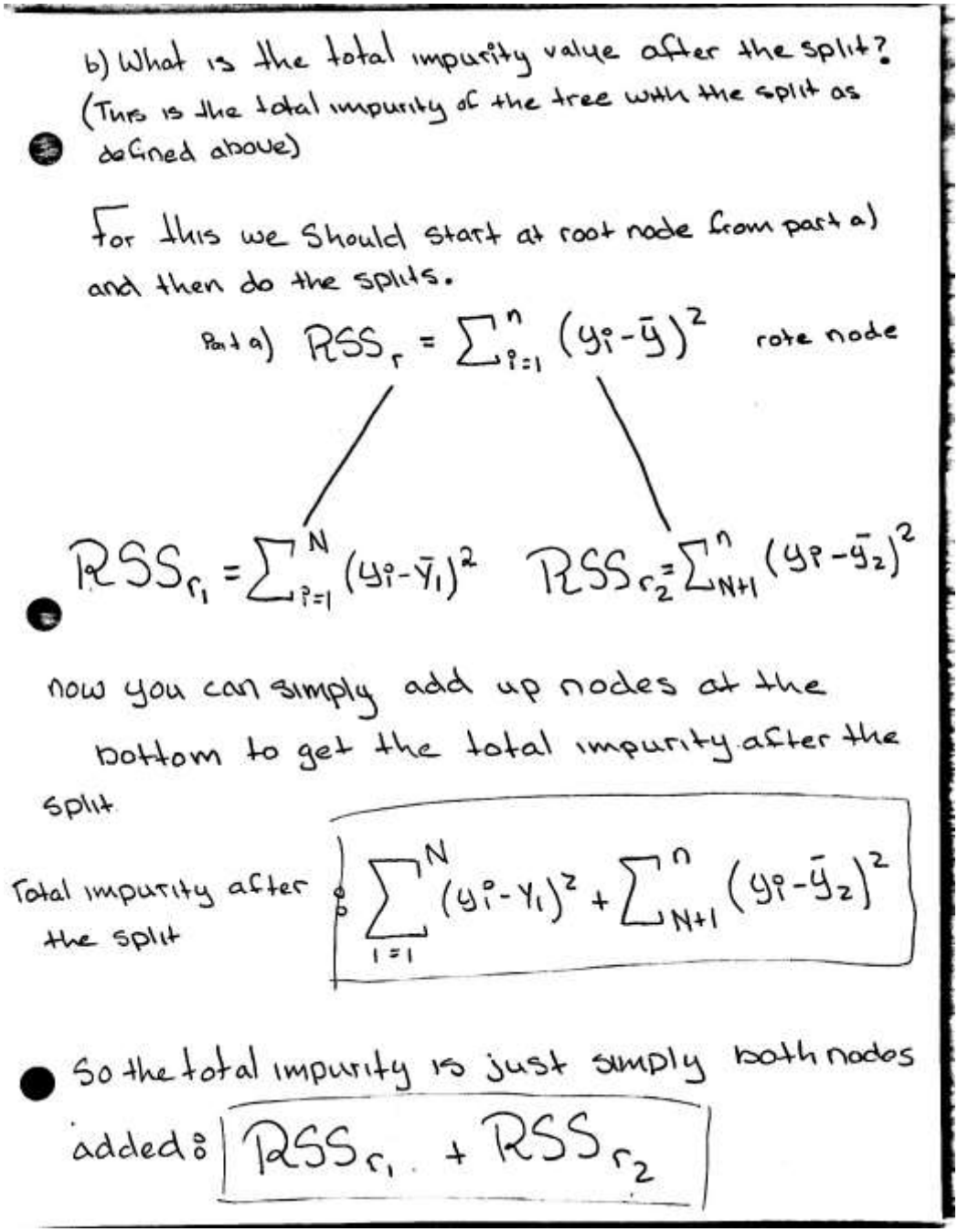
We can als write it in term of its node, which is root node:

$$RSS_r = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Someone mentioned the baseline model should have the value of only 0.5, but in lecture or GSI told me it should be the mean value based on how you write the RSS. Because I wrote it in terms of Y_hat which says, you are predicting the value for any given value to be the mean value for the baseline model.

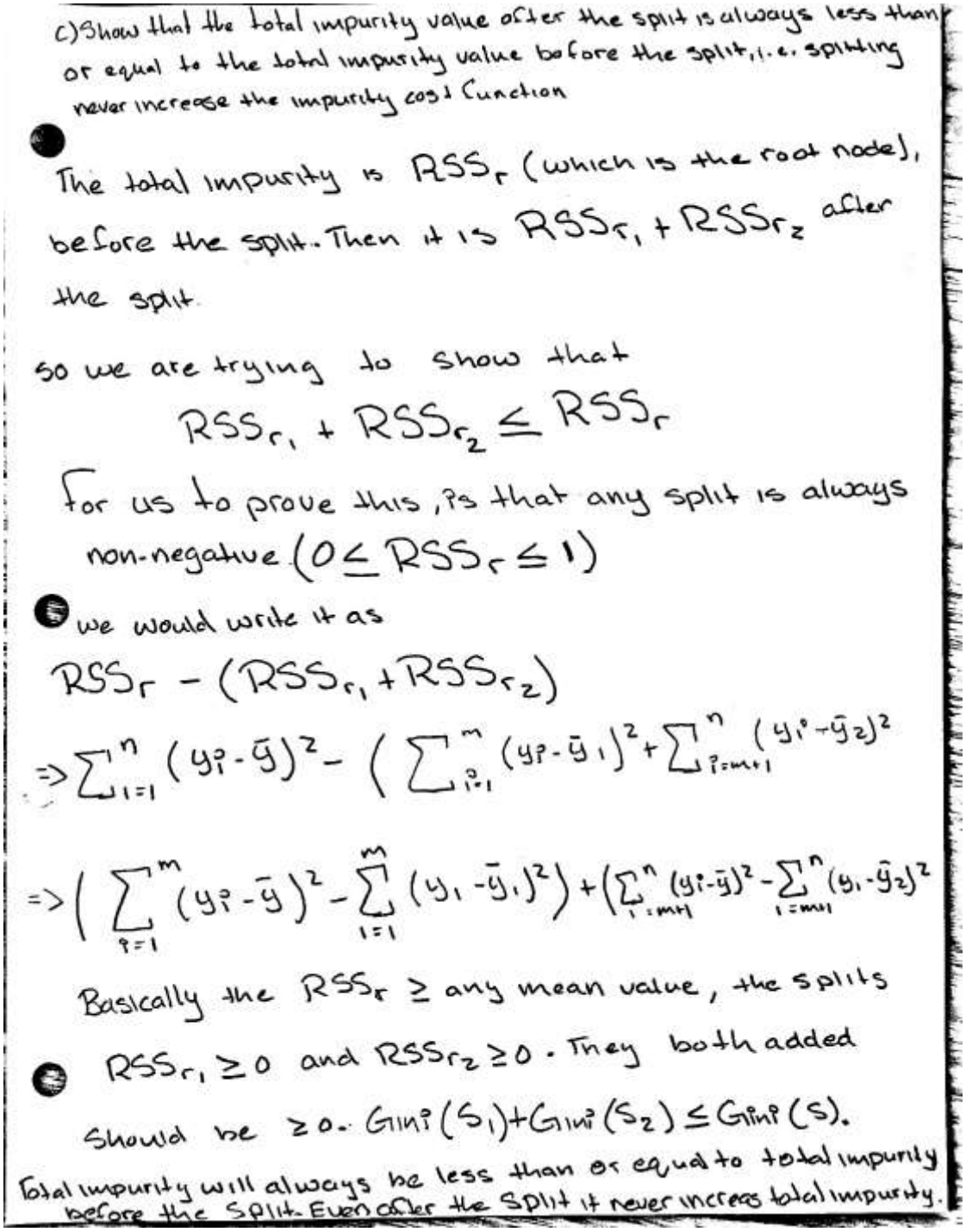
Part 1.b

b) (5 points) What is the total impurity value after the split? (This is the total impurity of the tree with the split as defined above.)



Part 1.C

c) (10 points) Show that the total impurity value after the split is always less than or equal to the total impurity value before the split, i.e., splitting never increases the total impurity cost function. (Hint: you can use the fact that, given a sequence of real numbers z1, z2, . . . , zn, the mean ¯z = 1 n Pn i=1 zi is the minimizer of the function RSS(z) = Pn i=1(zi − z) 2 .)



The total impurity will always be less than or equal to the total impurity before the split, even after the split it never increase the total impurity. Basically they should always be less than or equal to 0 for the left side and the right side should also be less than or eqal to 0, and both of them combined should be less than or equal to 0.

In [ ]:

As mentioned, the above reasoning applies to an arbitrary split of the training dataset into a parti□tion of size 2. The CART algorithm only considers splits of a particular type – those corresponding to two regions R1(j, s) = {X : Xj < s} and R2(j, s) = {X : Xj ≥ s} where j is the index of a chosen feature and s is a cutoff value.

d) (10 points) Consider a modification of the regression tree algorithm such that, in addition to considering splits of the form described in the preceding paragraph, we also consider splits of the form R1(j, t) = {X : e Xj < t} and R2(j, t) = {X : e Xj ≥ t} where j is the index of a chosen feature and t is a cutoff value for the exponential function e Xj . Is it possible for these new splits to improve the regression tree? Explain

In [3]:
```
import pandas as pd
from sklearn.metrics import accuracy_score
import numpy as np

data = {
    'Temperature (C)': [5, 10, 15, 20, 25, 30, 35, 40],
    'Day Type': ['Cold', 'Cold', 'Cold', 'Cold', 'Hot', 'Hot', 'Hot', 'Hot']
}

df = pd.DataFrame(data)
feature_column = 'Temperature (C)'
best_threshold_hot = None
best_accuracy_hot = 0.0
best_threshold_cold = None
best_accuracy_cold = 0.0

for threshold in df[df['Day Type'] == 'Hot'][feature_column]:
    predictions = (df[df['Day Type'] == 'Hot'][feature_column] > threshold).ast
    accuracy = accuracy_score(df[df['Day Type'] == 'Hot']['Day Type'].map({'Hot
    if accuracy > best_accuracy_hot:
        best_accuracy_hot = accuracy
        best_threshold_hot = threshold

for threshold in df[df['Day Type'] == 'Cold'][feature_column]:
    predictions = (df[df['Day Type'] == 'Cold'][feature_column] > threshold).as
    accuracy = accuracy_score(df[df['Day Type'] == 'Cold']['Day Type'].map({'Ho
    if accuracy > best_accuracy_cold:
        best_accuracy_cold = accuracy
        best_threshold_cold = threshold

print("Best Threshold (Without Transformation):", best_threshold_hot)
print("Best Accuracy (Without Transformation):", best_accuracy_hot)

exponential_values_hot = np.exp(df[df['Day Type'] == 'Hot'][feature_column])
best_threshold_hot = None
best_accuracy_hot = 0.0

for threshold in exponential_values_hot:
    predictions = (exponential_values_hot > threshold).astype(int)
    accuracy = accuracy_score(df[df['Day Type'] == 'Hot']['Day Type'].map({'Hot
    if accuracy > best_accuracy_hot:
        best_accuracy_hot = accuracy
        best_threshold_hot = threshold

best_threshold_cold = None
best_accuracy_cold = 0.0

for threshold in df[df['Day Type'] == 'Cold'][feature_column]:
    predictions = (df[df['Day Type'] == 'Cold'][feature_column] > threshold).as
    accuracy = accuracy_score(df[df['Day Type'] == 'Cold']['Day Type'].map({'Ho
    if accuracy > best_accuracy_cold:
        best_accuracy_cold = accuracy
        best_threshold_cold = threshold

print("Best Threshold (With Transformation):", best_threshold_cold)
print("Best Accuracy (With Transformation):", best_accuracy_cold)
```

```
Best Threshold (Without Transformation): 25
Best Accuracy (Without Transformation): 0.75
Best Threshold (With Transformation): 20
Best Accuracy (With Transformation): 1.0
```

Answer 1.D

For my example, I did simple coding example, where I am doing Cold vs Hot day. So lets look at the first case, where you do the threshold value for R1(j, s) = {X : Xj < s} and R2(j, s) = {X : Xj ≥ s}, you get the values, for threshold as 25 Celcius, and accracy for 0.75. Then I do the transformation which the question asked, R1(j, t) = {X : e Xj < t} and R2(j, t) = {X : e Xj ≥ t} where the value goes down for the thresdhold to 20, and the accracy increases all the way to 1.0. This shows that using exponential of given values improves the model. We went from 0.75 to 1.0 for predicting what type of day it is. This shows how you can improve the splits of regression tree by taking the exponential value of given column.

I should address that I used very specific type of data numbers for this to work, as it was an example to prove by taking expoenntial values you can improve the model because when I did random generator data for hot and cold days, it would make the accracy worse by adding in the exponentail values. The conclusion here is, that it won't work in every situation, you can only improve the tree regression splits based on the dataset you have, by having exponential values as it creates extreme as you see the threshold value dropped by 5 celicus but increased the accracy all they way to 1.0. The exponential values seem to work when the data ins't linear, it has to be non-linear, so by taking exponential valeus, you make big difference between value, hence making better threshold value to classify the split of given node.

e) (10 points) Consider a modification of the regression tree algorithm such that, in addition to considering splits of the form described in the paragraph preceding part (d), we also consider splits of the form R1(j, ℓ, t) = {X : XjXℓ < t} and R2(j, ℓ, t) = {X : XjXℓ ≥ t} where j and ℓ are the indices of two chosen features and t is a cutoff value for XjXℓ . Is it possible for these new splits to improve the regression tree? Explain.

In [4]:

```
import pandas as pd
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix

# Load the dataset
CHD_train = pd.read_csv("framingham_train.csv")

# Step 1: Predict using a single feature (BMI)
feature_name = "BMI"
X = CHD_train[[feature_name]]
y = CHD_train["TenYearCHD"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random

# Create a decision tree regressor
reg_tree = DecisionTreeRegressor(random_state=42)
reg_tree.fit(X_train, y_train)

# Predict
y_pred = reg_tree.predict(X_test)

# Find the threshold for TPR and FPR
threshold_values = [0.1, 0.2, 0.3, 0.4, 0.5]
results_single_feature = []

for threshold in threshold_values:
    y_pred_binary = (y_pred >= threshold).astype(int)
    tn, fp, fn, tp = confusion_matrix(y_test, y_pred_binary).ravel()
    tpr = tp / (tp + fn)
    fpr = fp / (fp + tn)
    accuracy = accuracy_score(y_test, y_pred_binary)
    results_single_feature.append((threshold, tpr, fpr, accuracy))

print("Results for single feature:")
print("Threshold\tTPR\tFPR\tAccuracy")
for result in results_single_feature:
    threshold, tpr, fpr, accuracy = result
    print(f"{threshold:.1f}\t\t{tpr:.2f}\t{fpr:.2f}\t{accuracy:.2f}")

# Step 2: Predict using a feature interaction (BMI * cigsPerDay)
feature1_name = "BMI"
feature2_name = "cigsPerDay"
X = CHD_train[[feature1_name, feature2_name]]
X["interaction"] = X[feature1_name] * X[feature2_name]
y = CHD_train["TenYearCHD"]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random

# Create a decision tree regressor
reg_tree = DecisionTreeRegressor(random_state=42)
reg_tree.fit(X_train, y_train)

# Predict
```

```
y_pred = reg_tree.predict(X_test)

# Find the threshold for TPR and FPR
results_feature_interaction = []

for threshold in threshold_values:
    y_pred_binary = (y_pred >= threshold).astype(int)
    tn, fp, fn, tp = confusion_matrix(y_test, y_pred_binary).ravel()
    tpr = tp / (tp + fn)
    fpr = fp / (fp + tn)
    accuracy = accuracy_score(y_test, y_pred_binary)
    results_feature_interaction.append((threshold, tpr, fpr, accuracy))

print("\nResults for feature interaction:")
print("Threshold\tTPR\tFPR\tAccuracy")
for result in results_feature_interaction:
    threshold, tpr, fpr, accuracy = result
    print(f"{threshold:.1f}\t\t{tpr:.2f}\t{fpr:.2f}\t{accuracy:.2f}")
```

```
Results for single feature:
Threshold       TPR     FPR     Accuracy
0.1             0.23    0.27    0.65
0.2             0.22    0.25    0.68
0.3             0.18    0.19    0.71
0.4             0.15    0.16    0.74
0.5             0.14    0.15    0.75

Results for feature interaction:
Threshold       TPR     FPR     Accuracy
0.1             0.19    0.17    0.74
0.2             0.19    0.17    0.74
0.3             0.19    0.16    0.74
0.4             0.19    0.14    0.76
0.5             0.19    0.14    0.76

C:\Users\jackf\AppData\Local\Temp\ipykernel_12716\2778499124.py:46: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  X["interaction"] = X[feature1_name] * X[feature2_name]
```

Answer 1.E

In this example to prove how R1(j, ℓ, t) = {X : XjXℓ < t} and R2(j, ℓ, t) = {X : XjXℓ ≥ t} are an improvement over R1(j, s) = {X : Xj < s} and R2(j, s) = {X : Xj ≥ s}, I did the example from the last homework dataeset of framingham. Here I made the Xj to be just BMI to predict on TenYearCHD, and you can see in the chart above the highest threshold hold value is 0.75 with TPR 0.14 and FPR of 0.15. Then I did the second model where I did XJXL where I

did Xj as BMI and XL as cigsperday to predict on TenYearCHD, you can see the results go up by little bit, the accracy improved to 0.76 and the TPR increased to 0.19 and FPR to 0.14, so the model overall is doing better than orignal mode. However, you can notice that threshold 0.1, the accracy was only 0.65 for BMI but for BMI * cigsperday it was at 0.74, a very big improvement. This example showed how multiplying two features together improved the model and reduced the threshold value for the regression tree.

Simarily to question 1.D it should be noted that it doesn't work with all features, some features by themselves are better at splitting for example age but if you add any variable it decreases the value. So I did a lot of running and testing to where I picked BMI and cigsperday to show that it can indeed improve the regression tree splits by multiplying both variables, but doesn't work in all cases. It will depend on the datasets themselves, but more on the feature ineteractions with one another, where if they are able to improve the model or complexity of the features where they might be too linear and multicolinearity happens causing them to overfitting and mispredicting.

In [ ]:

Problem 2: Predicting Yelp Ratings (60 points)

Yelp is a widely popular platform that publishes information and reviews of local businesses such as restaurants, plumbers, hair salons, and others. Any user of Yelp is able to write a review, and each review includes a star rating between 1 and 5 in addition to written comments. In this problem, you will build models for predicting the star ratings of restaurants in Las Vegas, Nevada based on attributes contained in their Yelp profiles. Such a model may be useful for businesses to understand which factors are most important in attaining a high star rating and in gaining popularity more generally. The data for this problem is contained in the files yelp142 train.csv and yelp142 test.csv, and was retrieved from the larger Yelp Dataset1 provided by Yelp. In particular, we will focus our analysis on a subset of the Yelp data concerning restaurants in the Las Vegas, Nevada area. We have performed a random 70/30 split, resulting in 6,272 observations in the training set and 2,688 observations in the test set. Each observation contains the average star rating, number of reviews, and a list of attributes collected from the Yelp page of a particular restaurant in the Las Vegas area. These attributes are described in Table 1. Note that variable selection is not required for this problem.

NOTE: Whenever a question asks you to perform some coding task, such as building a model, please include an explanation of what the code does (e.g., "Below is the code for building a linear regression model") as well as the supporting code.

Part 1.A

a) (5 points) There are many missing entries in this dataset, denoted by (Missing) in the data files. In particular, all of the attribute features contain missing values and Table 1 reports the percentage of observations where each attribute is missing. In general, there are several approaches for dealing with missing values in supervised learning. Each attribute with missing values in our dataset is a categorical feature and, in the subsequent models that you will build, you should treat (Missing) as an explicit category. Do you think this modeling approach is reasonable or not? Explain your answer

Yes this reasonable, to treat the missing entries of the dataset as a explicity category. One reason is if we drop the missing values, we may loss on important data or some sort of anaylsis from it. I recently did titantic dataset, where dropping missing values, would result in bad regression because people that had missing data correlated highly with death as result of the tragic incident. So dropping any missing value, can mean the resturant themselves didn't have good data collecting skills or simply are just bad resturants, and we shouldn't drop their data values. This would make our prediction off if I stand right by my assumption. Also we can make one hot encoder for missing values

Part 2.B

b) (20 points) Let us start by building regression models for predicting stars based on all of the provided features listed in Table 1. All of your models should, of course, be built only using the training data provided in the yelp142 train.csv file.

part i

i)First build a linear regression model. Remember to use all of the provided independent variables, and you do not have to do variable selection in this problem. For each of the categorical variables, you should use (Missing) as the reference level to be incorporated into the intercept term. This does not affect the predictive performance of the model,but it does lead to a cleaner interpretation. This can be achieved in statsmodels with a slight modification to the R-style formulas. For example, you could use code like "stars ~ review_count + C(GoodForKids, Treatment(reference='(Missing)'))" for a model regressing stars on review count and GoodForKids

I simply built the linear regression model using all the variable as the question had asked me to do so. I called all the variables on "stars", which would tell us how the star rating is affected by other variables, we should pick the ones with low variables if we want to continue to improve our model. I noticed that there a lot features that have small p-value, indicating they are important variables. However, the model is really bad as the adjusted r-squared value is only 0.170, telling me that we shouldn't do a linear regression but some other regression that will help us predict better for our model.

In [5]:
```
train = pd.read_csv("yelp142_train.csv")
linear_regression = smf.ols(formula =
      "stars ~ review_count \
    + C(GoodForKids, Treatment(reference='(Missing)')) \
    + C(Alcohol, Treatment(reference='(Missing)')) \
    + C(BusinessAcceptsCreditCards, Treatment(reference='(Missing)')) \
    + C(WiFi, Treatment(reference='(Missing)')) \
    + C(BikeParking, Treatment(reference='(Missing)')) \
    + C(ByAppointmentOnly, Treatment(reference='(Missing)')) \
    + C(WheelechairAccessible, Treatment(reference='(Missing)')) \
    + C(OutdoorSeating, Treatment(reference='(Missing)')) \
    + C(RestaurantsReservations, Treatment(reference='(Missing)')) \
    + C(DogsAllowed, Treatment(reference='(Missing)')) \
    + C(Caters, Treatment(reference='(Missing)'))", data=train).fit()

print(linear_regression.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  stars   R-squared:                       0.17
3
Model:                            OLS   Adj. R-squared:                  0.17
0
Method:                 Least Squares   F-statistic:                     52.3
8
Date:                Sat, 14 Oct 2023   Prob (F-statistic):          1.46e-23
5
Time:                        22:43:50   Log-Likelihood:                -7239.
0
No. Observations:                6272   AIC:                         1.453e+0
4
Df Residuals:                    6246   BIC:                         1.471e+0
4
Df Model:                          25
Covariance Type:            nonrobust
==============================================================================
==========================================================

coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
------------------------------------------------------------
Intercept
3.4062      0.040     84.470      0.000       3.327       3.485
C(GoodForKids, Treatment(reference='(Missing)'))[T.FALSE]
0.0196      0.045      0.434      0.664      -0.069       0.108
C(GoodForKids, Treatment(reference='(Missing)'))[T.TRUE]                    -
0.0966      0.035     -2.759      0.006      -0.165      -0.028
C(Alcohol, Treatment(reference='(Missing)'))[T.'beer_and_wine']
0.2451      0.048      5.129      0.000       0.151       0.339
C(Alcohol, Treatment(reference='(Missing)'))[T.'full_bar']
0.1200      0.044      2.732      0.006       0.034       0.206
C(Alcohol, Treatment(reference='(Missing)'))[T.'none']
0.1211      0.040      3.046      0.002       0.043       0.199
C(BusinessAcceptsCreditCards, Treatment(reference='(Missing)'))[T.FALSE]
0.4796      0.090      5.318      0.000       0.303       0.656
C(BusinessAcceptsCreditCards, Treatment(reference='(Missing)'))[T.TRUE]
0.0513      0.047      1.089      0.276      -0.041       0.144
C(WiFi, Treatment(reference='(Missing)'))[T.'free']
0.1519      0.034      4.438      0.000       0.085       0.219
C(WiFi, Treatment(reference='(Missing)'))[T.'no']
0.1468      0.033      4.442      0.000       0.082       0.212
C(WiFi, Treatment(reference='(Missing)'))[T.'paid']                         -
0.2846      0.110     -2.581      0.010      -0.501      -0.068
C(BikeParking, Treatment(reference='(Missing)'))[T.FALSE]                   -
0.2009      0.032     -6.284      0.000      -0.264      -0.138
C(BikeParking, Treatment(reference='(Missing)'))[T.TRUE]                    -
0.1200      0.029     -4.177      0.000      -0.176      -0.064
C(ByAppointmentOnly, Treatment(reference='(Missing)'))[T.FALSE]
0.1270      0.033      3.797      0.000       0.061       0.193
C(ByAppointmentOnly, Treatment(reference='(Missing)'))[T.TRUE]
0.2601      0.098      2.667      0.008       0.069       0.451
C(WheelechairAccessible, Treatment(reference='(Missing)'))[T.FALSE]
0.7377      0.092      8.011      0.000       0.557       0.918
```

```
C(WheelechairAccessible, Treatment(reference='(Missing)'))[T.TRUE]
0.3651      0.027     13.284      0.000       0.311       0.419
C(OutdoorSeating, Treatment(reference='(Missing)'))[T.FALSE]                -
0.1359      0.040     -3.389      0.001      -0.215      -0.057
C(OutdoorSeating, Treatment(reference='(Missing)'))[T.TRUE]                 -
0.0253      0.043     -0.591      0.555      -0.109       0.059
C(RestaurantsReservations, Treatment(reference='(Missing)'))[T.FALSE]       -
0.2205      0.041     -5.396      0.000      -0.301      -0.140
C(RestaurantsReservations, Treatment(reference='(Missing)'))[T.TRUE]        -
0.0077      0.045     -0.169      0.866      -0.096       0.081
C(DogsAllowed, Treatment(reference='(Missing)'))[T.FALSE]
0.2571      0.029      8.874      0.000       0.200       0.314
C(DogsAllowed, Treatment(reference='(Missing)'))[T.TRUE]
0.1737      0.054      3.189      0.001       0.067       0.280
C(Caters, Treatment(reference='(Missing)'))[T.FALSE]                        -
0.1250      0.030     -4.137      0.000      -0.184      -0.066
C(Caters, Treatment(reference='(Missing)'))[T.TRUE]
0.1057      0.033      3.233      0.001       0.042       0.170
review_count                                                              8.6
89e-05   2.59e-05      3.359      0.001    3.62e-05       0.000
==============================================================================
=
Omnibus:                      124.334   Durbin-Watson:                   2.01
0
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              131.56
2
Skew:                          -0.355   Prob(JB):                     2.70e-2
9
Kurtosis:                       2.979   Cond. No.                     5.40e+0
3
==============================================================================
=

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
[2] The condition number is large, 5.4e+03. This might indicate that there ar
e
strong multicollinearity or other numerical problems.
```

I simply built the linear regression model using all the variable as the question had asked me to do so. I called all the variables on "stars", which would tell us how the star rating is affected by other variables, we should pick the ones with low variables if we want to continue to improve our model. I noticed that there a lot features that have small p-value, indicating they are important variables. However, the model is really bad as the adjusted r-squared value is only 0.170, telling me that we shouldn't do a linear regression but some other regression that will help us predict better for our model.

Part II

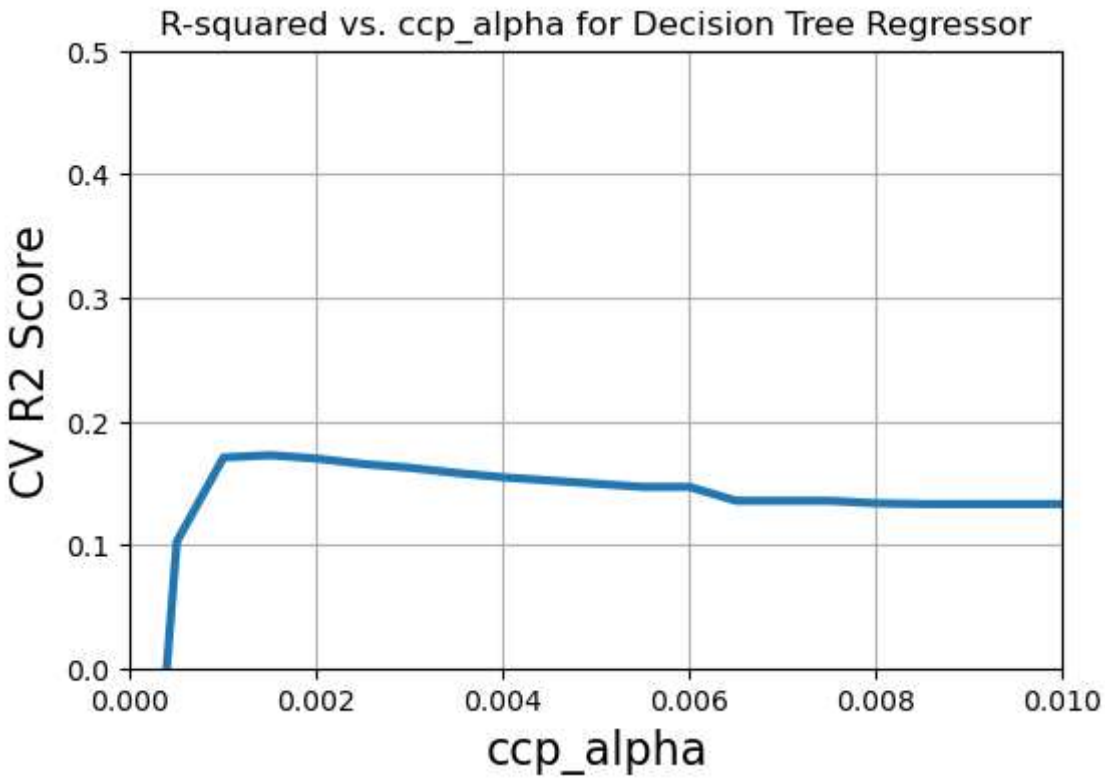
Now build a regression tree model (using an implementation of the CART algorithm). Select the complexity parameter (i.e., ccp alpha in sklearn) value for the tree through cross-validation, and explain how you did the cross-validation and how you selected the complexity parameter value.

For this code, I simply first made copies of the dataset, and then split them into y_train and y_test and X_train and X_test, while getting out the dummy variable because we need them categorical format of 0 and 1. The 0 and 1 value come from using .get_dummies code. Up next made variable to contain the ccp_alpha values, I am using the same values from the lab6. Then we make our model on decisionTreeRegressor, using random_state = 88, same as the lab to keep it consistent. Also this doesn't make a difference what random_state you use, you should get close to the same value, or evne the same value, just the values in between will be changed a little bit. Then we do the gridsearch, which takes in hyperparamater, in which I use the ccp_alpha values and r2 as my scoring, which is for getting the r-squared value. And then we also have k_fold = 5, which how many time we did cross validation. Then I plotted the graph which is the code from lab6, to see it visually, you can't really see but I adjuted the values a little bit where you can see it peak around 0.0015.

In [7]:
```
test = pd.read_csv("yelp142_test.csv")
yelp_train_copy = train.copy()
y_train = yelp_train_copy["stars"]
X_train = pd.get_dummies(yelp_train_copy.drop(["stars"], axis = 1))

yelp_test_copy = test.copy()
y_test = yelp_test_copy["stars"]
X_test = pd.get_dummies(yelp_test_copy.drop(["stars"], axis = 1))

grid_values = {'ccp_alpha': np.linspace(0, 0.10, 201)}
dtr = DecisionTreeRegressor(random_state = 88)
cv = KFold(n_splits=5, random_state = 88, shuffle=True)
decision_tree_reg = GridSearchCV(dtr, param_grid=grid_values, scoring = "r2", c
decision_tree_reg.fit(X_train, y_train)
print("Best ccp_alpha value is: ",decision_tree_reg.best_params_)
```

```
Best ccp_alpha value is:  {'ccp_alpha': 0.0015}
```

In [8]:
```
ccp_alpha = grid_values['ccp_alpha']
r2_scores = decision_tree_reg.cv_results_['mean_test_score']
plt.figure(figsize=(6, 4))
plt.plot(ccp_alpha, r2_scores, lw=3)
plt.xlabel('ccp_alpha', fontsize=16)
plt.ylabel('CV R2 Score', fontsize=16)
plt.grid(True, which='both')
plt.xlim([0, 0.01])
plt.ylim([0.0, 0.5])
plt.title("R-squared vs. ccp_alpha for Decision Tree Regressor")

plt.show()
```



We get the value of 0.0015, which is our best complexity parameter value. This basically is our highest value we got from the model we just built, it had others but 0.0015 was the best. Also you can see it in the graph the value peaks around 0.0015. This the maximized r squared value in our training data, so we should use this value when builting our model in the future.

part III

Using the test set data provided in the yelp142 test.csv file, compute the OSR2 values of your linear regression and regression tree models. Also, compute the MAE (mean absolute error) values of both models. How do you judge the performance of the two models?

I used the code from the lab to calculate the OSR2. The OSR is getting us the overall score of the regression model. We are testing if the linear regression model is better or the decision tree regression is model is better, hence you see linear.predict() and

dtr.predict(), and the MAE is getting the average absolute value error between the actual values and predicted values. But this code is to compare each other.

In [9]:
```
def OSR2(model, df_train, df_test, var):
    y_test = df_test[var]
    y_pred = model.predict(df_test)
    SSE = np.sum((y_test-y_pred)**2)
    SST = np.sum((y_test - np.mean(df_train[var]))**2)
    return (1 - SSE/SST)

def OSR2_DTR(model, X_test, y_test, y_train):
    y_pred = model.predict(X_test)
    SSE = np.sum((y_test - y_pred) ** 2)
    SST = np.sum((y_test - np.mean(y_train))**2)
    return (1 - SSE/SST)

print("Linear Regression OSR2",(OSR2(linear_regression, train, test, "stars")))
print("Dicision Tree Regression OSR2",(OSR2_DTR(decision_tree_reg, X_test, y_te

dtr_y_pred = decision_tree_reg.best_estimator_.predict(X_test)
lr_y_pred = linear_regression.predict(test)
lr_y_test = test["stars"]
print("Linear_Regression MAE", mean_absolute_error(lr_y_pred,lr_y_test))
print("Dicision TRee Regression MAE", mean_absolute_error(dtr_y_pred,lr_y_test)
```

```
Linear Regression OSR2 0.14799194607877697
Dicision Tree Regression OSR2 0.1504339434181572
Linear_Regression MAE 0.6259556143398647
Dicision TRee Regression MAE 0.617440936299048
```

The performance both models is just bad. and the values are OSR are 0.1479,0.1504 and MAE are 0.6259,0.617. These models are really bad, they both have low OSR2 score. Also should note that they aren't better than one another either, they both have value differing by only .1 in OSR2 and .1 in MAE as well, indicating they are just bad models, we should do something else to improve our prediction of the model.

Part 2.C

c) (5 points) Regression may not be the most appropriate modeling technique for this data. In particular, it is plausible that restaurants may be mostly concerned with ensuring that their star rating is high enough and not particularly concerned with precisely predicting the value of this rating. Therefore, let us instead consider a classification problem where the goal is to predict if the star rating is greater than or equal to 4 or not.

part I

i) Construct a new variable in your training and test datasets called fourOrAbove. This variable should be equal to 1 if stars is greater than or equal to 4 and equal to 0 otherwise.

This code is simply making a new column named fourOrAbove, where we making values based on stars, if they are greater than 4 stars they will be labled as 1 and if they are less than 4 stars they will be labeled as 0. This like creating dummie variable we could make through one_hot_encoder for a machine learning model to work better on it.

In [10]:
```
train["fourOrAbove"] = ([1 if x >= 4.0 else 0 for x in train["stars"]])
test["fourOrAbove"]  = ([1 if x >= 4.0 else 0 for x in test["stars"]])
```

Part 2.D

d)(30 points) Let us now work on building classification models for predicting fourOrAbove based on all of the provided features listed in Table 1. All of your models should, of course, be built only using the training data provided in the yelp142 train.csv file.

Part i)

i) In this problem, we will weigh false positives and false negatives equally and therefore focus on accuracy (equivalently, error rate) as the primary performance metric. Do you think this modeling choice is reasonable or not? Explain your answer.

Yes, it is reasonable to weigh the false positives and false negatives equally because we don't if one or the other will cause us an issue. This isn't like our last homework assignment where false and negative positives made a difference how much we charge for co-pay or premimum to our customers, so we should treat them equally unless we have to treat them differently where the values will matter to us. We should just focus on our accracy of our model rather than FP and NP

Part II

A simple approach here is to use the previously built linear regression and regression tree models to address the classification task by thresholding their predictions at the value of 4. Write code for implementing this thresholding procedure for converting the predictions of your previously built regression models to predictions of fourOrAbove

I created two variables, one for our linear regression model where we predicted on values greater than or equal to 4, and if the value is greater than 4 we give it a 1 and if its less than 4 we assign the value 0. Similar process happened on the other variable. This basically what the question asked us to do, to implement a code where thresholding for converting the predictions for our previous models on this near variable that we just had.

In [30]:
```
## new linear regression prediction and new diciesion tree regressions
new_lr_pred  = ([1 if x >= 4 else 0 for x in lr_y_pred])
new_dtr_pred = ([1 if x >= 4 else 0 for x in dtr_y_pred])
```

Part III

iii) Now build a logistic regression model. Remember to use all of the provided independent variables, and you do not have to do variable selection in this problem. Please again use (Missing) as the reference level to be incorporated into the intercept term

Similar to part A, this time we make logistic regression model with all the variables. We use our new train table which has values 4.0 and above. This model is looking better than our other model based on the current funcation value being around 0.60 where the other model only had 0.170. Basically just copied what I did in part A but this time changed word to logit for logistic regression.

In [12]:
```
## just making sure the values are correct still
train.head() ## just making sure the values are correct still
```

Out[12]:

	stars	review_count	GoodForKids	Alcohol	BusinessAcceptsCreditCards	WiFi	Bike
0	4.5	16	(Missing)	'none'	TRUE	'no'	
1	3.5	537	TRUE	'beer_and_wine'	TRUE	'no'	
2	3.0	444	TRUE	'full_bar'	TRUE	'no'	
3	3.0	4	(Missing)	(Missing)	(Missing)	(Missing)	
4	5.0	3	(Missing)	(Missing)	TRUE	'no'	

In [13]:
```
logistic_regression = smf.logit(formula =
      "fourOrAbove ~ review_count \
    + C(GoodForKids, Treatment(reference='(Missing)')) \
    + C(Alcohol, Treatment(reference='(Missing)')) \
    + C(BusinessAcceptsCreditCards, Treatment(reference='(Missing)')) \
    + C(WiFi, Treatment(reference='(Missing)')) \
    + C(BikeParking, Treatment(reference='(Missing)')) \
    + C(ByAppointmentOnly, Treatment(reference='(Missing)')) \
    + C(WheelechairAccessible, Treatment(reference='(Missing)')) \
    + C(OutdoorSeating, Treatment(reference='(Missing)')) \
    + C(RestaurantsReservations, Treatment(reference='(Missing)')) \
    + C(DogsAllowed, Treatment(reference='(Missing)')) \
    + C(Caters, Treatment(reference='(Missing)'))", data= train).fit()

print(logistic_regression.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.608425
         Iterations 6
                           Logit Regression Results
==============================================================================
=
Dep. Variable:            fourOrAbove   No. Observations:                  627
2
Model:                          Logit   Df Residuals:                      624
6
Method:                           MLE   Df Model:                            2
5
Date:                Sat, 14 Oct 2023   Pseudo R-squ.:                   0.113
5
Time:                        22:46:13   Log-Likelihood:                -3816.
0
converged:                       True   LL-Null:                       -4304.
8
Covariance Type:            nonrobust   LLR p-value:                 3.284e-19
0
==============================================================================
==============================================================

coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
------------------------------------------------------------
Intercept                                                                    -
0.3101      0.107     -2.890      0.004      -0.520      -0.100
C(GoodForKids, Treatment(reference='(Missing)'))[T.FALSE]
0.2469      0.128      1.934      0.053      -0.003       0.497
C(GoodForKids, Treatment(reference='(Missing)'))[T.TRUE]                     -
0.2024      0.099     -2.045      0.041      -0.396      -0.008
C(Alcohol, Treatment(reference='(Missing)'))[T.'beer_and_wine']
0.1748      0.137      1.273      0.203      -0.094       0.444
C(Alcohol, Treatment(reference='(Missing)'))[T.'full_bar']                   -
0.2038      0.127     -1.601      0.109      -0.453       0.046
C(Alcohol, Treatment(reference='(Missing)'))[T.'none']
0.3084      0.114      2.694      0.007       0.084       0.533
C(BusinessAcceptsCreditCards, Treatment(reference='(Missing)'))[T.FALSE]
1.0310      0.255      4.041      0.000       0.531       1.531
C(BusinessAcceptsCreditCards, Treatment(reference='(Missing)'))[T.TRUE]      -
0.0309      0.127     -0.244      0.807      -0.279       0.217
C(WiFi, Treatment(reference='(Missing)'))[T.'free']
0.4209      0.097      4.328      0.000       0.230       0.612
C(WiFi, Treatment(reference='(Missing)'))[T.'no']
0.3314      0.094      3.514      0.000       0.147       0.516
C(WiFi, Treatment(reference='(Missing)'))[T.'paid']                          -
0.7700      0.363     -2.123      0.034      -1.481      -0.059
C(BikeParking, Treatment(reference='(Missing)'))[T.FALSE]                    -
0.4982      0.092     -5.399      0.000      -0.679      -0.317
C(BikeParking, Treatment(reference='(Missing)'))[T.TRUE]                     -
0.3079      0.081     -3.800      0.000      -0.467      -0.149
C(ByAppointmentOnly, Treatment(reference='(Missing)'))[T.FALSE]
0.4357      0.097      4.512      0.000       0.246       0.625
C(ByAppointmentOnly, Treatment(reference='(Missing)'))[T.TRUE]
0.3991      0.290      1.374      0.169      -0.170       0.968
C(WheelechairAccessible, Treatment(reference='(Missing)'))[T.FALSE]
```

```
2.0350      0.338      6.021      0.000       1.373       2.697
C(WheelechairAccessible, Treatment(reference='(Missing)'))[T.TRUE]
0.9324      0.076     12.309      0.000       0.784       1.081
C(OutdoorSeating, Treatment(reference='(Missing)'))[T.FALSE]               -
0.3339      0.113     -2.956      0.003      -0.555      -0.113
C(OutdoorSeating, Treatment(reference='(Missing)'))[T.TRUE]                -
0.1566      0.120     -1.304      0.192      -0.392       0.079
C(RestaurantsReservations, Treatment(reference='(Missing)'))[T.FALSE]      -
0.4197      0.116     -3.631      0.000      -0.646      -0.193
C(RestaurantsReservations, Treatment(reference='(Missing)'))[T.TRUE]
0.0352      0.128      0.275      0.783      -0.215       0.286
C(DogsAllowed, Treatment(reference='(Missing)'))[T.FALSE]
0.7095      0.081      8.744      0.000       0.550       0.868
C(DogsAllowed, Treatment(reference='(Missing)'))[T.TRUE]
0.4271      0.156      2.730      0.006       0.120       0.734
C(Caters, Treatment(reference='(Missing)'))[T.FALSE]                       -
0.3458      0.086     -4.001      0.000      -0.515      -0.176
C(Caters, Treatment(reference='(Missing)'))[T.TRUE]
0.1160      0.092      1.261      0.207      -0.064       0.296
review_count
0.0005   9.74e-05      5.080      0.000       0.000       0.001
==============================================================================
==============================================================
```

Part IV

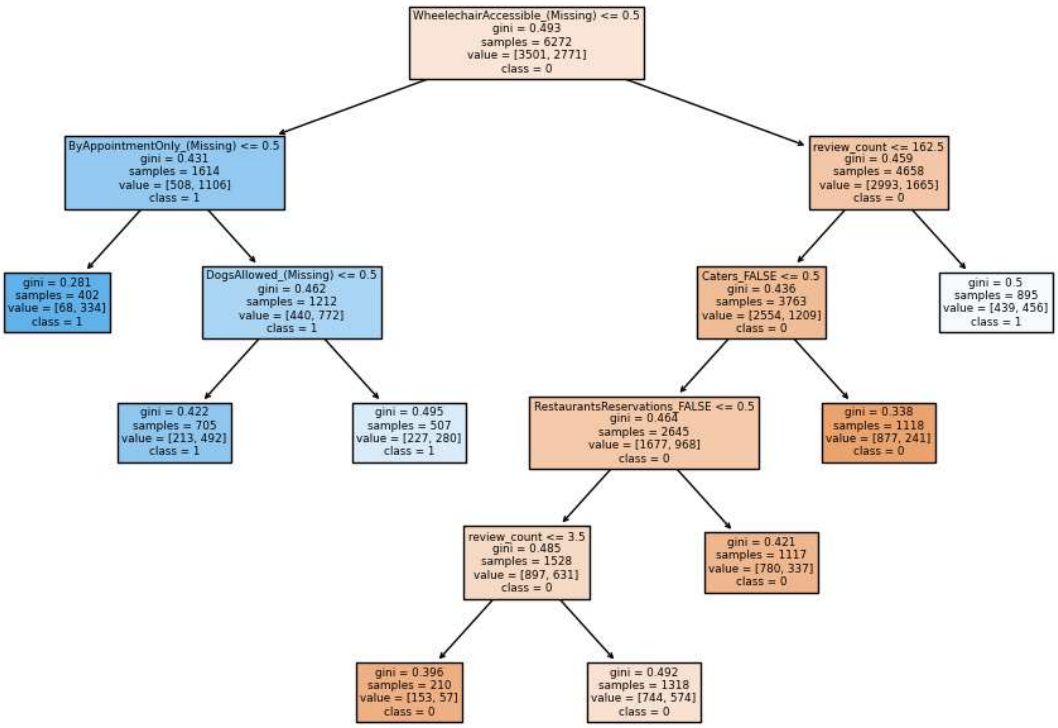
iv) Now build a classification tree model (using an implementation of the CART algorithm). Select the complexity parameter (i.e., ccp alpha in sklearn) value for the tree through cross-validation, and explain how you did the cross-validation and how you selected the complexity parameter value. Produce a diagram to show your decision tree.

This is the same code as previous code, just changed up the values from stars to four0rabove. But we are using the first half of the code to find the best ccp_alpha value. Then we use the tree to plot. I did max_leaf_nodes = 8, because if I do anymore you can barely see the tree, as it goes into a lot of depth, so having 8 nodes it perfect or even less. Also the code for that is from lab06, copied and put in the correct variables. Had to make y_train and X_train dummies for the model to work. Everything else is just fitted in. Again the random state doesn't matter, I went with 42, as its the most famous random_state but the value for ccp_alpha is lower than previous model or whenever we did it.

In [20]:
```
yelp_train = train.copy()
y_train = yelp_train["fourOrAbove"]
X_train = pd.get_dummies(yelp_train.drop(["fourOrAbove", "stars"], axis=1))


dt_classifier = DecisionTreeClassifier(random_state=42)
param_grid = {'ccp_alpha': np.linspace(0, 0.002, 101)}
grid_search = GridSearchCV(dt_classifier, param_grid, cv=5)
grid_search.fit(X_train, y_train)
best_ccp_alpha = grid_search.best_params_['ccp_alpha']
print(f"Best ccp_alpha: {best_ccp_alpha}")
```

```
Best ccp_alpha: 0.00042
```

In [28]:
```
best_tree = DecisionTreeClassifier(ccp_alpha=best_ccp_alpha, random_state=42, m
best_tree.fit(X_train, y_train)

feature_names = X_train.columns.tolist()
plt.figure(figsize=(12, 8))
plot_tree(best_tree, filled=True, feature_names=feature_names, class_names=["0'
plt.show()
```



As we can see the tree starts with wheelchariAcessible, that makes sense, if resturant isn't wheelchairacessible, that can probably indicate, they "aren't" fancy or not for everyone, so most people wouldn't want to go to resturants that are harder to be accessible, also you have to think about bathrooms and other stuff. The sample is only 1660, so they are not lossing on much.On the left node it splits it up by appointpment, that also makes sense in its kid node which is dogsallowed, because if you make an

appointment, you most likely will ask them if you can bring your dogs or not. I think this

Part v)

v) Produce a table for evaluating the performance of the models that you have built. The table should consider the following models: a baseline model that predicts the most frequent outcome for fourOrAbove, the two regression models with thresholding, and all of the classification models built in this part of the problem. Additionally, the table should report the following performance metrics: accuracy (our primary performance metric), TPR, and FPR. All of these performance metrics should be computed using the test set data provided in the yelp142 test.csv file. Do the results seem reasonable to you? How do you judge the performance of the models and the trade-offs between different models? Which model would you recommend for this problem and why

Here I am making accracy and TPR and FPR score using the confusion matrix, and filling the values correctly for each model, not much you can explain about the code. I did set the baseline model == 0 and == 1, as that is baseline model and used the values that we generated in above models before this.

In [98]:
```
base_pred = np.sum(test["fourOrAbove"] == 0)
base_pred_1 = np.sum(test["fourOrAbove"] == 1)
accruacy = base_pred/ (base_pred + base_pred_1)
TPR = 0
FPR = 0
print("Baseline Mode")
print("Accuracy:", accruacy)
print("TPR", TPR)
print("FPR", FPR)
```

```
Baseline Mode
Accuracy: 0.5598958333333334
TPR 0
FPR 0
```

In [103]:
```
cm = confusion_matrix(y_test,new_lr_pred)
print("Linear Regression")
print(cm)

acc = (cm.ravel()[0] + cm.ravel()[3])/sum(cm.ravel())
tpr = cm.ravel()[3]/ (cm.ravel()[3] + cm.ravel()[2])
fpr = cm.ravel()[1]/ (cm.ravel()[1] + cm.ravel()[0])
print("Accuracy", acc)
print("TPR", tpr)
print("FPR", fpr)
```

```
Linear Regression
[[1447   58]
 [ 930  253]]
Accuracy 0.6324404761904762
TPR 0.21386306001690616
FPR 0.038538205980066444
```

In [104]:
```
# Decision Tree model
cm = confusion_matrix(y_test, new_dtr_pred )
print("Confusion Matrix")
print(cm)

acc = (cm.ravel()[0] + cm.ravel()[3])/sum(cm.ravel())
tpr = cm.ravel()[3]/ (cm.ravel()[3] + cm.ravel()[2])
fpr = cm.ravel()[1]/ (cm.ravel()[1] + cm.ravel()[0])
print("Accuracy", acc)
print("TPR", tpr)
print("FPR", fpr)
```

```
Confusion Matrix
[[1447   58]
 [ 947  236]]
Accuracy 0.6261160714285714
TPR 0.19949281487743026
FPR 0.038538205980066444
```

In [105]:
```
# Logistic Regression Model
y_prob = logistic_regression.predict(test)
y_pred = pd.Series([1 if x > 0.5 else 0 for x in y_prob], index = y_prob.index)

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix")
print(cm)

acc = (cm.ravel()[0] + cm.ravel()[3])/sum(cm.ravel())
tpr = cm.ravel()[3]/ (cm.ravel()[3] + cm.ravel()[2])
fpr = cm.ravel()[1]/ (cm.ravel()[1] + cm.ravel()[0])
print("Accuracy", acc)
print("TPR", tpr)
print("FPR", fpr)
```

```
Confusion Matrix
[[1250  255]
 [ 616  567]]
Accuracy 0.6759672619047619
TPR 0.47928994082840237
FPR 0.16943521594684385
```

In [106]:
```
#Decision Tree classifer model
yelp_test = test.copy()
y_text = yelp_test["fourOrAbove"]
X_test = pd.get_dummies(yelp_test.drop(["fourOrAbove","stars"], axis = 1))
y_pred = dicision_tree_classifier.predict(X_test)

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix")
print(cm)

acc = (cm.ravel()[0] + cm.ravel()[3])/sum(cm.ravel())
tpr = cm.ravel()[3]/ (cm.ravel()[3] + cm.ravel()[2])
fpr = cm.ravel()[1]/ (cm.ravel()[1] + cm.ravel()[0])
print("Accuracy", acc)
print("TPR", tpr)
print("FPR", fpr)
```

```
Confusion Matrix
[[755 750]
 [423 760]]
Accuracy 0.5636160714285714
TPR 0.6424344885883347
FPR 0.4983388704318937
```

In [29]:
```
data = [["Baseline model",0.55,0,0],
        ["Linear Regression model",0.63, 0.21,0.038],
        ["Dicion Tree Regression model", 0.62,0.199,0.038 ],
        ["Logistic Regression model",  0.675, 0.479, 0.169],
        ["Decision Tree Classifer model",0.563,0.642,0.498]]
dataFrame = pd.DataFrame(data,columns = ["Model type", "Accuracy", "TPR", "FPR"
dataFrame
```

Out[29]:

	Model type	Accuracy	TPR	FPR
0	Baseline model	0.550	0.000	0.000
1	Linear Regression model	0.630	0.210	0.038
2	Dicion Tree Regression model	0.620	0.199	0.038
3	Logistic Regression model	0.675	0.479	0.169
4	Decision Tree Classifer model	0.563	0.642	0.498

Answer: The results are resonable to me, I assumed all the values would be really close to one another, and baseline would be bad. Seems like logistic regression did the best in terms of accuracy, so I would recommend that model for this problem simply because it did the best, but if we care about higher TPR or FPR I would recommend Decision Tree Classifier even if it has lower accuracy then lgositic regression. I think it depends on what the person is looking for, like we talked about in part i, if we care for TPR and FPR, and if we do which one do you care more about, if you care just about having higher TPR and low FPR you would go with logistic regression model which also has a high accracy. So basically it comes down to what the person is trying to do but I would just stick with

logistic regression, since it has high accracy and we don't care for TPR and FPR.

In [ ]: