IEOR 242 HW3P1 Solution

Hyungki Im

Sep 2021

Problem 1(30 points)

Problem (a) (10 points)

WLOG, we assume that node M has been splitted. Then we have

$$\tilde{N}_m \tilde{Q}_m(T_{\text{new}}) = N_m Q_m(T_{\text{old}}), \ \forall m = 1, \cdots, M-1. \tag{1}$$

So we can simplify Δ as

$$\begin{aligned} \Delta &= N_M Q_M(T_{\text{old}}) - \big(\tilde{N}_M \tilde{Q}_M T(\text{new}) + \tilde{N}_{M+1} \tilde{Q}_{M+1} T(\text{new})\big) & (2)\\ &= \sum_{i:x_i \in R_M} (y_i - \hat{y}_M)^2 - \big(\sum_{i:x_i \in \tilde{R}_M} (y_i - \hat{\tilde{y}}_M)^2 + \sum_{i:x_i \in \tilde{R}_{M+1}} (y_i - \hat{\tilde{y}}_{M+1})^2 \big), & (3) \end{aligned}$$

where each $\tilde{R}_M$ and $\tilde{R}_{M+1}$ denotes the newly created region in T_{new} that satisfies $R_M = \tilde{R}_M \cup \tilde{R}_{M+1}$ and $\hat{\tilde{y}}_j = \frac{1}{\tilde{N}_j} \sum_{i:x_i \in \tilde{R}_j} y_i$, for $j = M, M+1$.

Grading Rubrics

Each sub question follows this grading rule.

- (-3) Missing (1)
- (-3) Missing (2)
- (-2) Missing (3)
- (-2) Missing definition of $\tilde{R}_M$ and $\hat{\tilde{y}}_j$

Problem (b) (10 points)

Lets define a function $RSS_A(z) := \sum_{i:i\in A}(y_i - z)^2$. As $R_M = \tilde{R}_M \cup \tilde{R}_{M+1}$, we can rewrite (3) as

$$\begin{aligned}
\Delta = &\sum_{i:x_i\in R_M}(y_i - \hat{y}_M)^2 - \Big(\sum_{i:x_i\in \tilde{R}_M}(y_i - \hat{\tilde{y}}_M)^2 + \sum_{i:x_i\in \tilde{R}_{M+1}}(y_i - \hat{\tilde{y}}_{M+1})^2\Big) && (4)\\
= &\sum_{i:x_i\in \tilde{R}_M}(y_i - \hat{y}_M)^2 + \sum_{i:x_i\in \tilde{R}_{M+1}}(y_i - \hat{y}_M)^2 && (5)\\
&- \Big(\sum_{i:x_i\in \tilde{R}_M}(y_i - \hat{\tilde{y}}_M)^2 + \sum_{i:x_i\in \tilde{R}_{M+1}}(y_i - \hat{\tilde{y}}_{M+1})^2\Big) && (6)\\
= &\Big(\sum_{i:x_i\in \tilde{R}_M}(y_i - \hat{y}_M)^2 - \sum_{i:x_i\in \tilde{R}_M}(y_i - \hat{\tilde{y}}_M)^2\Big) && (7)\\
&+ \Big(\sum_{i:x_i\in \tilde{R}_{M+1}}(y_i - \hat{y}_M)^2 - \sum_{i:x_i\in \tilde{R}_{M+1}}(y_i - \hat{\tilde{y}}_{M+1})^2\Big) && (8)\\
= &\big(RSS_{\tilde{R}_M}(\hat{y}_M) - RSS_{\tilde{R}_M}(\hat{\tilde{y}}_M)\big) + \big(RSS_{\tilde{R}_{M+1}}(\hat{y}_{M+1}) - RSS_{\tilde{R}_{M+1}}(\hat{\tilde{y}}_{M+1})\big) && (9)\\
\geq &0. && (10)
\end{aligned}$$

The last inequality comes from the hint.

Grading Rubrics

Each sub question follows this grading rule.

- (-5) Missing (5)
- (-5) Missing (10)
- This solution is not the only solution for this question. So when you think your solution is correct, give yourself full credit, otherwise deduct 2 points for each missing and miscalculation.

Problem (c) (10 points)

Using definition of $C_\alpha(T)$, we get

$$\begin{aligned}
&C_\alpha(T_{\text{new}}) - C_\alpha(T_{\text{old}}) \leq 0 && (11)\\
\iff &\sum_{\tilde{m}=1}^{M+1}\sum_{i:x_i\in \tilde{R}_{\tilde{m}}}(y_i - \hat{y}_{\tilde{m}})^2 - \sum_{m=1}^{M}\sum_{i:x_i\in R_m}(y_i - \hat{y}_m)^2 + \alpha SST \leq 0. && (12)
\end{aligned}$$

Then by definition of SSE, we have

$$\begin{aligned}
\iff &SSE_{\text{new}} - SSE_{\text{old}} \geq \alpha SST && (13)\\
\iff &\frac{SSE_{\text{new}}}{SST} - \frac{SSE_{\text{old}}}{SST} \geq \alpha && (14)\\
\iff &R^2_{\text{new}} - R^2_{\text{old}} \geq \alpha. && (15)
\end{aligned}$$

Grading Rubrics

Each sub question follows this grading rule.

- (-3) Missing (12)
- (-3) Missing (13)
- (-4) Missing (15)
- This solution is not the only solution for this question. So when you think your solution is correct, give yourself full credit, otherwise deduct 2 points for each missing and miscalculation.

IEOR242_F21_HW3P2_Solution

October 11, 2021

1 HW3 Solution and Code

1.1 By Hyungki Im

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
yelp_train = pd.read_csv("yelp242_train.csv")
yelp_test = pd.read_csv("yelp242_test.csv")

yelp_train.head(5)
```

[2]:
```
   stars  review_count GoodForKids          Alcohol  \
0    4.5           153       FALSE  'beer_and_wine'
1    3.5            19        TRUE        (Missing)
2    4.5             3        TRUE       'full_bar'
3    4.0           775        TRUE           'none'
4    3.5            24        TRUE       'full_bar'

  BusinessAcceptsCreditCards       WiFi BikeParking ByAppointmentOnly  \
0                       TRUE     'free'       FALSE         (Missing)
1                       TRUE     'free'   (Missing)         (Missing)
2                       TRUE  (Missing)   (Missing)         (Missing)
3                       TRUE     'free'        TRUE             FALSE
4                       TRUE     'free'   (Missing)         (Missing)

  WheelechairAccessible OutdoorSeating RestaurantsReservations DogsAllowed  \
0             (Missing)          FALSE                    TRUE       FALSE
1             (Missing)      (Missing)                   FALSE   (Missing)
2             (Missing)           TRUE               (Missing)   (Missing)
3             (Missing)           TRUE                    TRUE   (Missing)
4             (Missing)          FALSE                    TRUE   (Missing)

      Caters
0      FALSE
1  (Missing)
2  (Missing)
```

```
3       TRUE
4   (Missing)
```

2 Problem (a) (5 points)

This modeling choice is reasonable. There might be some pattern for '(missing)' independent variables. For example, quality of restaurant might have some relationship to the amount of information the restuarant provided. So instead of removing the data with missing values, we can treat (missing) as an explicit category.

2.1 Grading Rubrics

- (-2) Answered as unreasonable.
- (-3) Explanation is weak. No explanation of why '(missing)' could be a new catergorical level.

3 Problem (b) (15 points)

[3]:
```
yelp_train = pd.read_csv("yelp242_train.csv")
yelp_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6272 entries, 0 to 6271
Data columns (total 13 columns):
stars                         6272 non-null float64
review_count                  6272 non-null int64
GoodForKids                   6272 non-null object
Alcohol                       6272 non-null object
BusinessAcceptsCreditCards    6272 non-null object
WiFi                          6272 non-null object
BikeParking                   6272 non-null object
ByAppointmentOnly             6272 non-null object
WheelechairAccessible         6272 non-null object
OutdoorSeating                6272 non-null object
RestaurantsReservations       6272 non-null object
DogsAllowed                   6272 non-null object
Caters                        6272 non-null object
dtypes: float64(1), int64(1), object(11)
memory usage: 637.1+ KB
```

[4]:
```
yelp_train.head(5)
```

[4]:
```
   stars  review_count GoodForKids          Alcohol  \
0    4.5           153       FALSE  'beer_and_wine'
1    3.5            19        TRUE        (Missing)
2    4.5             3        TRUE       'full_bar'
3    4.0           775        TRUE           'none'
```

```
4    3.5            24        TRUE       'full_bar'

  BusinessAcceptsCreditCards       WiFi BikeParking ByAppointmentOnly  \
0                       TRUE     'free'       FALSE         (Missing)
1                       TRUE     'free'   (Missing)         (Missing)
2                       TRUE  (Missing)   (Missing)         (Missing)
3                       TRUE     'free'        TRUE             FALSE
4                       TRUE     'free'   (Missing)         (Missing)

  WheelechairAccessible OutdoorSeating RestaurantsReservations DogsAllowed  \
0             (Missing)          FALSE                    TRUE       FALSE
1             (Missing)      (Missing)                   FALSE   (Missing)
2             (Missing)           TRUE               (Missing)   (Missing)
3             (Missing)           TRUE                    TRUE   (Missing)
4             (Missing)          FALSE                    TRUE   (Missing)

      Caters
0      FALSE
1  (Missing)
2  (Missing)
3       TRUE
4  (Missing)
```

[5]:
```
#First Build Linear Regression Model
import statsmodels.formula.api as smf

model1_ols = 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 = yelp_train).fit()
model1_ols.summary()
```

[5]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  stars   R-squared:                       0.173
Model:                            OLS   Adj. R-squared:                  0.170
Method:                 Least Squares   F-statistic:                     52.33
Date:                Mon, 11 Oct 2021   Prob (F-statistic):          2.45e-235
Time:                        11:39:56   Log-Likelihood:                -7220.7
No. Observations:                6272   AIC:                         1.449e+04
Df Residuals:                    6246   BIC:                         1.467e+04
Df Model:                          25
Covariance Type:            nonrobust
================================================================================
============================================================
coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
------------------------------------------------------------
Intercept
3.3413      0.039     85.518      0.000       3.265       3.418
C(GoodForKids, Treatment(reference='(Missing)'))[T.FALSE]
-0.0329      0.046     -0.720      0.472      -0.123       0.057
C(GoodForKids, Treatment(reference='(Missing)'))[T.TRUE]
-0.1325      0.035     -3.757      0.000      -0.202      -0.063
C(Alcohol, Treatment(reference='(Missing)'))[T.'beer_and_wine']
0.1925      0.047      4.110      0.000       0.101       0.284
C(Alcohol, Treatment(reference='(Missing)'))[T.'full_bar']
0.1173      0.043      2.698      0.007       0.032       0.203
C(Alcohol, Treatment(reference='(Missing)'))[T.'none']
0.0921      0.039      2.363      0.018       0.016       0.169
C(BusinessAcceptsCreditCards, Treatment(reference='(Missing)'))[T.FALSE]
0.6324      0.087      7.257      0.000       0.462       0.803
C(BusinessAcceptsCreditCards, Treatment(reference='(Missing)'))[T.TRUE]
0.1338      0.046      2.897      0.004       0.043       0.224
C(WiFi, Treatment(reference='(Missing)'))[T.'free']
0.0685      0.034      1.998      0.046       0.001       0.136
C(WiFi, Treatment(reference='(Missing)'))[T.'no']
0.0858      0.033      2.594      0.009       0.021       0.151
C(WiFi, Treatment(reference='(Missing)'))[T.'paid']
-0.2794      0.103     -2.701      0.007      -0.482      -0.077
C(BikeParking, Treatment(reference='(Missing)'))[T.FALSE]
-0.1784      0.032     -5.608      0.000      -0.241      -0.116
C(BikeParking, Treatment(reference='(Missing)'))[T.TRUE]
-0.1117      0.029     -3.891      0.000      -0.168      -0.055
C(ByAppointmentOnly, Treatment(reference='(Missing)'))[T.FALSE]
0.1495      0.034      4.447      0.000       0.084       0.215
C(ByAppointmentOnly, Treatment(reference='(Missing)'))[T.TRUE]
0.2560      0.106      2.410      0.016       0.048       0.464
C(WheelechairAccessible, Treatment(reference='(Missing)'))[T.FALSE]
```

```
0.6685      0.090      7.468      0.000      0.493      0.844
C(WheelechairAccessible, Treatment(reference='(Missing)'))[T.TRUE]
0.3469      0.028     12.591      0.000      0.293      0.401
C(OutdoorSeating, Treatment(reference='(Missing)'))[T.FALSE]
-0.0755      0.040     -1.908      0.056     -0.153      0.002
C(OutdoorSeating, Treatment(reference='(Missing)'))[T.TRUE]
0.0168      0.042      0.399      0.690     -0.066      0.099
C(RestaurantsReservations, Treatment(reference='(Missing)'))[T.FALSE]
-0.2180      0.040     -5.451      0.000     -0.296     -0.140
C(RestaurantsReservations, Treatment(reference='(Missing)'))[T.TRUE]
-0.0084      0.045     -0.187      0.851     -0.096      0.079
C(DogsAllowed, Treatment(reference='(Missing)'))[T.FALSE]
0.2539      0.029      8.725      0.000      0.197      0.311
C(DogsAllowed, Treatment(reference='(Missing)'))[T.TRUE]
0.1346      0.054      2.516      0.012      0.030      0.239
C(Caters, Treatment(reference='(Missing)'))[T.FALSE]
-0.0840      0.030     -2.796      0.005     -0.143     -0.025
C(Caters, Treatment(reference='(Missing)'))[T.TRUE]
0.1657      0.033      5.096      0.000      0.102      0.229
review_count
0.0001   2.88e-05      3.566      0.000   4.63e-05      0.000
==============================================================================
Omnibus:                      130.621   Durbin-Watson:                   1.990
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              138.449
Skew:                          -0.363   Prob(JB):                     8.63e-31
Kurtosis:                       3.058   Cond. No.                     4.88e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.88e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[6]:
```
def OSR2(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)
```

[7]:
```
y_test = yelp_test['stars']
X_test = yelp_test.drop(['stars'], axis =1)

y_train = yelp_train['stars']
```

```
print('OSR2:', round(OSR2(model1_ols, X_test, y_test, y_train), 5))
```

```
OSR2: 0.15269
```

3.1 Problem (i) (5 points)

Above code shows us the implementation of linear regression to our dataset. We used smf.ols function to implement linear regression.

3.1.1 Grading Rubrics

- For this question, you don't have to present your linear regression model in a nice form
- (-2) Did not use '(Missing)' as the reference level to be incorporated into the intercept term. If you used C(GoodForKids, Treatment(reference='(Missing)')) for each independent variables, then you would gain full credit.
- (-2) Did not implement linear regression
- (-1) Missing independent variables. Note that in this problem set you should use all the independent variables that are in our dataset.

[8]:
```
#Construct CART
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor

#One hot encoding
y_train = yelp_train['stars']
X_train_dtr = pd.get_dummies(yelp_train.drop(['stars'], axis = 1))
```

[9]:
```
grid_values = {'ccp_alpha': np.linspace(0, 0.1, 200)}

dtr = DecisionTreeRegressor(min_samples_leaf = 5, min_samples_split=␣
↪20,random_state = 88)
dtr_cv = GridSearchCV(dtr, param_grid = grid_values, scoring = 'r2', cv = 5,␣
↪verbose = 0)
dtr_cv.fit(X_train_dtr,y_train)
```

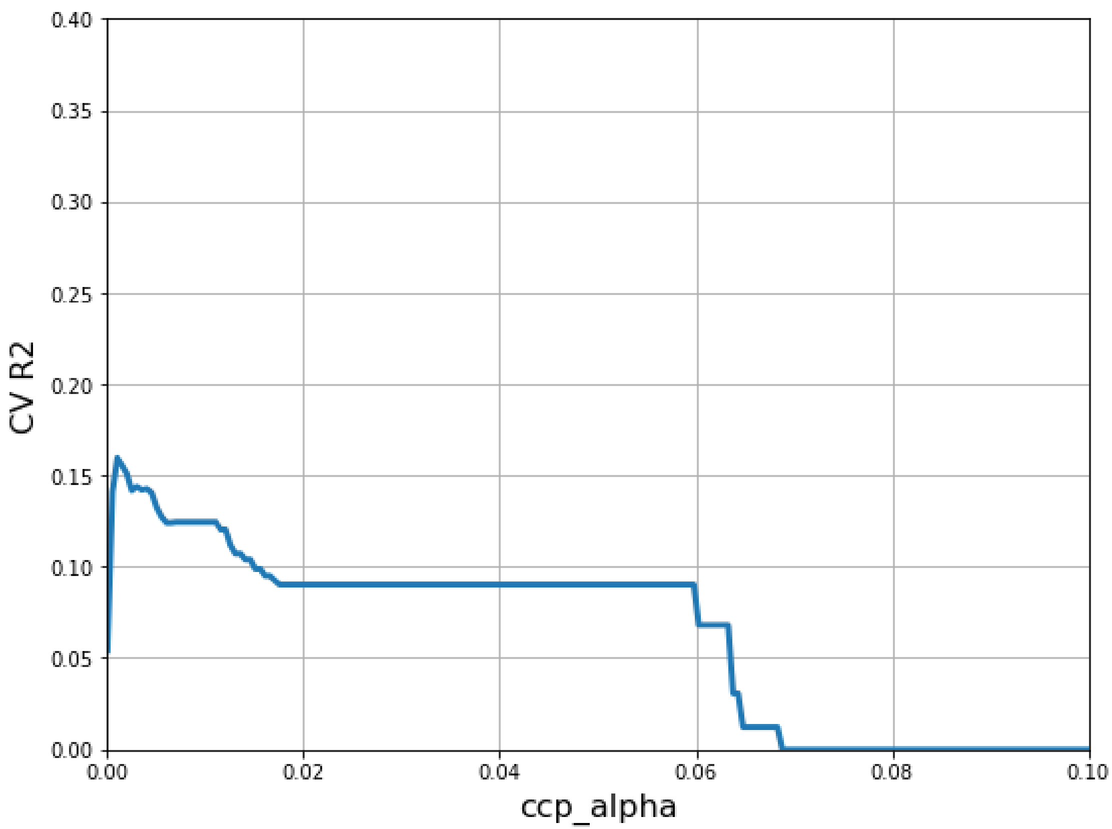
[9]:
```
GridSearchCV(cv=5,
             estimator=DecisionTreeRegressor(min_samples_leaf=5,
                                             min_samples_split=20,
                                             random_state=88),
             param_grid={'ccp_alpha': array([0.        , 0.00050251, 0.00100503,
0.00150754, 0.00201005,
       0.00251256, 0.00301508, 0.00351759, 0.0040201 , 0.00452261,
       0.00502513, 0.00552764, 0.00603015, 0.00653266, 0.00703518,
       0.00753769, 0.0080402 , 0.00854271, 0.00904523, 0.00954774,
       0…
       0.08291457, 0.08341709, 0.0839196 , 0.08442211, 0.08492462,
       0.08542714, 0.08592965, 0.08643216, 0.08693467, 0.08743719,
       0.0879397 , 0.08844221, 0.08894472, 0.08944724, 0.08994975,
```

```
0.09045226, 0.09095477, 0.09145729, 0.0919598 , 0.09246231,
0.09296482, 0.09346734, 0.09396985, 0.09447236, 0.09497487,
0.09547739, 0.0959799 , 0.09648241, 0.09698492, 0.09748744,
0.09798995, 0.09849246, 0.09899497, 0.09949749, 0.1       ])},
      scoring='r2')
```

[10]:
```python
#Plot our results of CV of CART
ccp_alpha = dtr_cv.cv_results_['param_ccp_alpha'].data
R2_scores = dtr_cv.cv_results_['mean_test_score']

plt.figure(figsize = (8,6))
plt.xlabel('ccp_alpha',fontsize =16)
plt.ylabel('CV R2', fontsize = 16)
plt.plot(ccp_alpha, R2_scores, linewidth=3)
plt.grid(True, which='both')
plt.xlim([0, 0.1])
plt.ylim([0.0, 0.4])

plt.tight_layout()
plt.show()
```



[11]:
```
y_test = yelp_test['stars']
X_test_dtr = pd.get_dummies(yelp_test.drop(['stars'], axis = 1))

print('Cross-validated R2:', round(dtr_cv.best_score_, 5))
print('OSR2:', round(OSR2(dtr_cv.best_estimator_, X_test_dtr, y_test, y_train),␣
↪5))
```

```
Cross-validated R2: 0.15949
OSR2: 0.1773
```

[18]:
```
print('Best ccp_alpha', dtr_cv.best_params_)
```

```
Best ccp_alpha {'ccp_alpha': 0.0010050251256281408}
```

3.2 Problem (ii) (5 points)

Above code shows the implementation of CART with cv. We used GridSearchCV function with DecisionTreeRegressor estimator and implemented 5-fold cross-validation. We looked up cp_value from 0 to 0.1 and draw the plot of our result. The plot shows that our search interval of cp_value are actually good. We will use best estimator(0.001) from our cross-validation.

3.2.1 Grading Rubrics

- (-2) Did not created dummy variables for our categorical variables. Either one-hot or dummy encoding is okay.
- (-2) Did not implement cross validation in appropriate way. One example of this case might be doing gridsearch over cp_values from 0.5 to 1.0 which is too big in our case.
- (-1) No explanation for their complexitiy parameter selection. Some ways to parameter selection could be following 1 standard rule or just choosing parameter that gives us the best result. In my code, I choosed parameter that returns the best result.

[12]:
```
#Calculate MAE and OSR2 and construct a table
from sklearn.metrics import mean_absolute_error

comparison_data = {'Linear Regression':['{:.3f}'.format(OSR2(model1_ols,␣
↪X_test, y_test, y_train)),
                                '{:.3f}'.
↪format(mean_absolute_error(y_test,model1_ols.predict(X_test)))],
                  'Regression Tree':['{:.3f}'.format(OSR2(dtr_cv.
↪best_estimator_, X_test_dtr, y_test, y_train)),
                              '{:.3f}'.
↪format(mean_absolute_error(y_test,dtr_cv.best_estimator_.
↪predict(X_test_dtr)))]}
comparison_table = pd.DataFrame(data = comparison_data, index = ['OSR2','MAE'])
comparison_table.style.set_properties(**{'font-size': '12pt',}).
↪set_table_styles([{'selector': 'th', 'props': [('font-size', '10pt')]}])
comparison_table.transpose()
```

[12]:

```
                     OSR2    MAE
Linear Regression  0.153  0.640
Regression Tree    0.177  0.621
```

3.3 Problem (iii) (5 points)

The above table shows us the desired result. We can say that both models did not perform well as both models' OSR2 is less then 0.2 and MAE is larger than 0.6.

3.3.1 Grading Rubrics

- (-2) Missing values of either MAE or OSR2.
- (-3) Did not judge the performance of the two models

[19]:

```python
#changing our response variable to categorical response variable
fourOrAbove_train = (yelp_train['stars'] >=4.0).astype(int)
fourOrAbove_test = (yelp_test['stars'] >=4.0).astype(int)

yelp_train_new = yelp_train.copy()
yelp_test_new = yelp_test.copy()

yelp_train_new['stars'] = fourOrAbove_train
yelp_train_new.rename(columns = {'stars':'fourOrAbove'}, inplace = True)
yelp_test_new['stars'] = fourOrAbove_test
yelp_test_new.rename(columns = {'stars':'fourOrAbove'},inplace =True)
```

3.3.2 Problem (c) (5 points)

3.3.3 Grading Rubrics

- (-5) Did not changed our dependent variable
- You don't have to display your result

4 Problem (d) (30 points)

4.1 Problem (i) (5 points)

There is no reason for this modeling choice to be unreasonable. We don't know how our model would be used exactly. So in this case, it is reasonable to assume that FP and FN has an equal cost.

4.1.1 Grading Rubrics

- (-5) Did not present reasonable answer. We would generously grade this question.

[21]:

```python
#Thresholding procedure of linear regression and regression tree model

ols_pred = (model1_ols.predict(X_test)>=4).astype(int)
dtr_pred = (dtr_cv.best_estimator_.predict(X_test_dtr)>=4).astype(int)
```

```
ols_pred.head(5)
```

[21]:
```
0    1
1    0
2    0
3    0
4    0
dtype: int32
```

4.2 Problem (ii) (5 points)

4.2.1 Grading Rubrics

- (-2) Wrong thresholding for linear regression
- (-3) Wrong thresholding for Decision Tree regression

[22]:
```
logreg = 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 = yelp_train_new).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.604809
         Iterations 6
```

[ ]:
```

```

4.3 Problem (iii) (5 points)

Above code shows how we constructed logistic regression. We used smf.logit function to implement logistic regression.

4.3.1 Grading Rubrics

- (-2) Did not set 'Missing' as an reference.
- (-3) Wrong logistic regression model

[23]:
```
#One hot encoding
y_train_new = yelp_train_new['fourOrAbove'].astype('int64')
y_test_new = yelp_test_new['fourOrAbove'].astype('int64')

X_train_new = yelp_train_new.drop(['fourOrAbove'],axis =1)
X_train_dtc = pd.get_dummies(X_train_new)

X_test_new = yelp_test_new.drop(['fourOrAbove'],axis =1)
X_test_dtc = pd.get_dummies(X_test_new)
```

[24]:
```
#Construct CART model
from sklearn.tree import DecisionTreeClassifier


grid_values = {'ccp_alpha': np.linspace(0, 0.1, 50)}

dtc = DecisionTreeClassifier(min_samples_leaf = 5, min_samples_split= 20,␣
↪max_depth = 30, \
                             random_state = 88)
dtc_cv = GridSearchCV(dtc, param_grid = grid_values, scoring = 'accuracy', cv =␣
↪5, verbose = 0)
dtc_cv.fit(X_train_dtc,y_train_new)
```

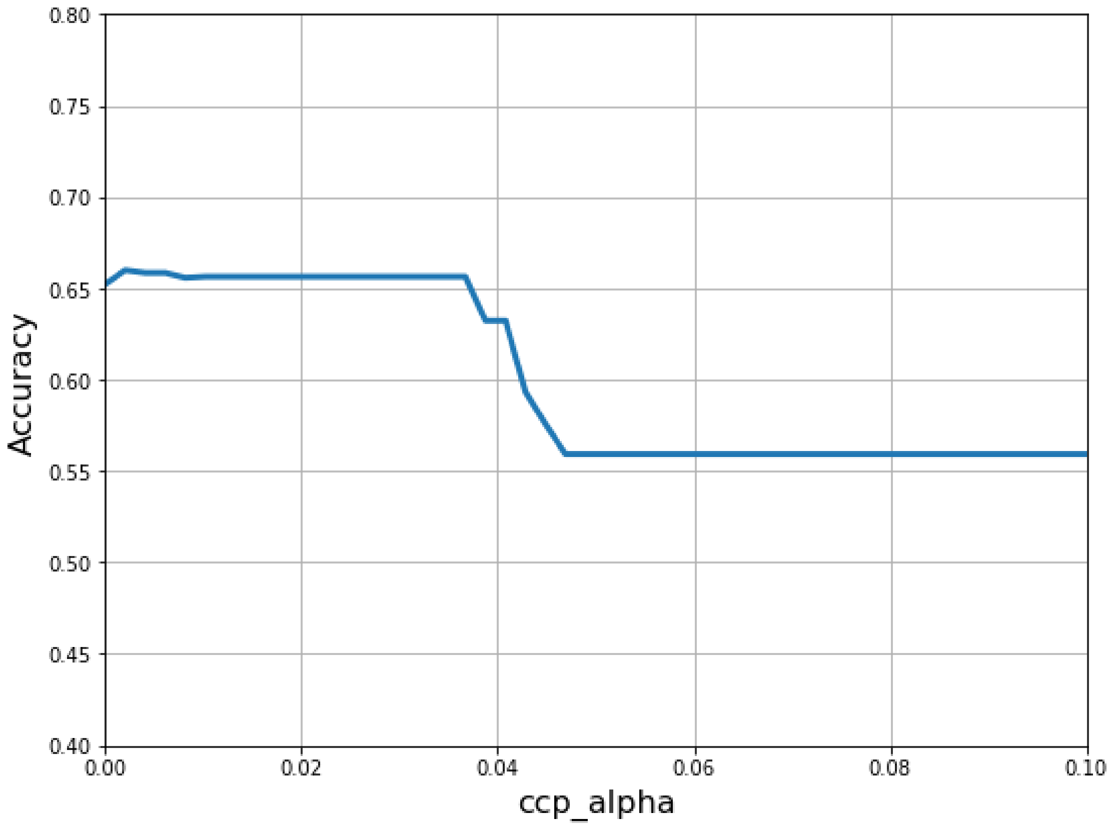
[24]:
```
GridSearchCV(cv=5,
             estimator=DecisionTreeClassifier(max_depth=30, min_samples_leaf=5,
                                              min_samples_split=20,
                                              random_state=88),
             param_grid={'ccp_alpha': array([0.        , 0.00204082, 0.00408163,
0.00612245, 0.00816327,
       0.01020408, 0.0122449 , 0.01428571, 0.01632653, 0.01836735,
       0.02040816, 0.02244898, 0.0244898 , 0.02653061, 0.02857143,
       0.03061224, 0.03265306, 0.03469388, 0.03673469…877551,
       0.04081633, 0.04285714, 0.04489796, 0.04693878, 0.04897959,
       0.05102041, 0.05306122, 0.05510204, 0.05714286, 0.05918367,
       0.06122449, 0.06326531, 0.06530612, 0.06734694, 0.06938776,
       0.07142857, 0.07346939, 0.0755102 , 0.07755102, 0.07959184,
       0.08163265, 0.08367347, 0.08571429, 0.0877551 , 0.08979592,
       0.09183673, 0.09387755, 0.09591837, 0.09795918, 0.1       ])},
             scoring='accuracy')
```

[25]:
```
#Plot our results of CV of CART
dtc_ccp_alpha = dtc_cv.cv_results_['param_ccp_alpha'].data
dtc_acc = dtc_cv.cv_results_['mean_test_score']

plt.figure(figsize = (8,6))
plt.xlabel('ccp_alpha',fontsize =16)
plt.ylabel('Accuracy', fontsize = 16)
```

```
plt.plot(dtc_ccp_alpha, dtc_acc, linewidth=3)
plt.grid(True, which='both')
plt.xlim([0, 0.1])
plt.ylim([.4, 0.8])

plt.tight_layout()
plt.show()
```



4.4 Problem (iv) (7 points)

Above code shows how we constructed decision tree classifier with 5-fold cross validation. We implemented cross validation over parameter ‘ccp_alpha’ which takes value from 0 to 0.1. We used ‘accuracy’ as our evaluation metric. As it will be shown in problem (v) we pick the cp value that returns the best accuracy.

4.4.1 Grading Rubrics

- (-2) Used decision tree regressor instead of classifier.
- (-1) Did not explain how did they implemented cross validation.
- (-2) Did not explain how did you selected the complexity parameter.
- (-2) Wrong construction of DTC.

[26]:
```python
#Let's build baseline model
default_false = np.sum(yelp_train_new['fourOrAbove']==0)
default_true = np.sum(yelp_train_new['fourOrAbove']==1)

print(pd.Series({'0': default_false, '1': default_true}))
```

```
0    3508
1    2764
dtype: int64
```

[27]:
```python
#Statistics of baseline model
from sklearn.metrics import confusion_matrix

baseline_acc = default_false/(default_true+default_false)
baseline_TPR = 0
baseline_FPR = 0
```

[28]:
```python
#Statistics of linear regression model
cm = confusion_matrix(y_test_new, ols_pred)
print ("Confusion Matrix : \n", cm)

lin_acc = (cm.ravel()[0]+cm.ravel()[3])/sum(cm.ravel())
lin_TPR = cm.ravel()[3]/(cm.ravel()[2]+cm.ravel()[3])
lin_FPR = cm.ravel()[1]/(cm.ravel()[0]+cm.ravel()[1])
```

```
Confusion Matrix :
 [[1436   62]
 [ 981  209]]
```

[29]:
```python
#Statistics of DecisionTreeRegressor

cm = confusion_matrix(y_test_new, dtr_pred)
print ("Confusion Matrix : \n", cm)

dtr_acc = (cm.ravel()[0]+cm.ravel()[3])/sum(cm.ravel())
dtr_TPR = cm.ravel()[3]/(cm.ravel()[2]+cm.ravel()[3])
dtr_FPR = cm.ravel()[1]/(cm.ravel()[0]+cm.ravel()[1])
```

```
Confusion Matrix :
 [[1445   53]
 [ 991  199]]
```

[30]:
```python
#Statistics of logistic regression model

log_prob = logreg.predict(yelp_test_new)
log_pred = pd.Series([1 if x > 0.5 else 0 for x in log_prob], index=log_prob.
 ↪index)
```

```
cm = confusion_matrix(y_test_new, log_pred)
print ("Confusion Matrix : \n", cm)

log_acc = (cm.ravel()[0]+cm.ravel()[3])/sum(cm.ravel())
log_TPR = cm.ravel()[3]/(cm.ravel()[2]+cm.ravel()[3])
log_FPR = cm.ravel()[1]/(cm.ravel()[0]+cm.ravel()[1])
```

```
Confusion Matrix :
 [[1225  273]
 [ 623  567]]
0.6666666666666666
0.4764705882352941
0.1822429906542056
```

[31]:
```
#Statistics of Decision Tree Classifier

dtc_pred = dtc_cv.best_estimator_.predict(X_test_dtc)

cm = confusion_matrix(y_test_new, dtc_pred)
print ("Confusion Matrix : \n", cm)

dtc_acc = (cm.ravel()[0]+cm.ravel()[3])/sum(cm.ravel())
dtc_TPR = cm.ravel()[3]/(cm.ravel()[2]+cm.ravel()[3])
dtc_FPR = cm.ravel()[1]/(cm.ravel()[0]+cm.ravel()[1])
```

```
Confusion Matrix :
 [[1203  295]
 [ 617  573]]
```

[36]:
```
#Now let's construct comparison table
comparison_data = {'Baseline':[baseline_acc,baseline_TPR,baseline_FPR],'Linear␣
↪Regression with Thresholding': [lin_acc,lin_TPR,lin_FPR],
                    'Decision Tree Regressor with Thresholding':
↪[dtr_acc,dtr_TPR,dtr_FPR], 'Logistic Regression':[log_acc,\
                     log_TPR,log_FPR],'Decision Tree Classifier':
↪[dtc_acc,dtc_TPR,dtc_FPR]}

comparison_table = pd.DataFrame(data=comparison_data, index=['Accuracy', 'TPR',␣
↪'FPR']).transpose()
comparison_table.style.set_properties(**{'font-size': '12pt',}).
↪set_table_styles([{'selector': 'th', 'props': [('font-size', '10pt')]}])
comparison_table
```

[36]:

	Accuracy	TPR	FPR
Baseline	0.559311	0.000000	0.000000
Linear Regression with Thresholding	0.611979	0.175630	0.041389

Decision Tree Regressor with Thresholding	0.611607	0.167227	0.035381
Logistic Regression	0.666667	0.476471	0.182243
Decision Tree Classifier	0.660714	0.481513	0.196929

4.5 Problem (v) (8 points)

Above table summarizes our results. The best two model that performs well in terms of our primary metric ‘accuracy’ are logistic regression and DTC. Theses two models dominates other models in accuracy and in TPR. This result seems reasonable as the both models are actually designed for classification while the other two are designed for regresion. If I have to choose one of these models, I would choose logistic regressoin as accuracy is our primary evaluation metric.

4.5.1 Grading Rubrics

- (-1) Did not use test set for evaluation.
- (-1) Any missing cell would be deducted 1 point each and up to total 5 points. If students did not present as an table, they would be deducted 5 points.
- (-2) No reasonable comparison between the models.

[43]:
```python
from sklearn.ensemble import RandomForestClassifier
import time

num_features = X_train_new.shape[1]
grid_values = {'max_features': np.linspace(1,num_features,num_features,␣
 ↪dtype='int32'),
               'min_samples_leaf': [5],
               'n_estimators': [500],
               'random_state': [88]}

tic = time.time()

rf2 = RandomForestClassifier()
rf_cv = GridSearchCV(rf2, param_grid=grid_values, scoring='accuracy', cv=5)
rf_cv.fit(X_train_dtc, y_train_new)

toc = time.time()

print('time:', round(toc-tic, 2),'s')
```

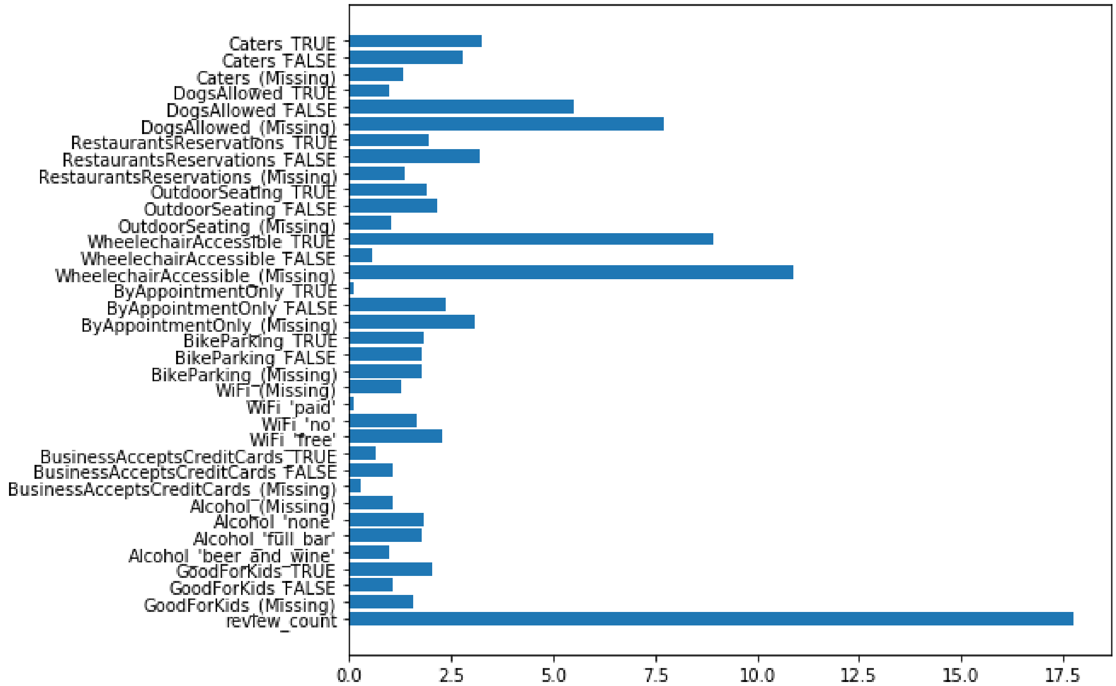
```
time: 465.08 s
```

[44]:
```python
pd.DataFrame({'Feature' : X_train_dtc.columns,
              'Importance score': 100*rf_cv.best_estimator_.
 ↪feature_importances_}).round(1)
```

[44]:

	Feature	Importance score
0	review_count	17.8
1	GoodForKids_(Missing)	1.6
2	GoodForKids_FALSE	1.1
3	GoodForKids_TRUE	2.0
4	Alcohol_'beer_and_wine'	1.0
5	Alcohol_'full_bar'	1.8
6	Alcohol_'none'	1.8
7	Alcohol_(Missing)	1.1
8	BusinessAcceptsCreditCards_(Missing)	0.3
9	BusinessAcceptsCreditCards_FALSE	1.1
10	BusinessAcceptsCreditCards_TRUE	0.6
11	WiFi_'free'	2.3
12	WiFi_'no'	1.7
13	WiFi_'paid'	0.1
14	WiFi_(Missing)	1.3
15	BikeParking_(Missing)	1.8
16	BikeParking_FALSE	1.8
17	BikeParking_TRUE	1.8
18	ByAppointmentOnly_(Missing)	3.1
19	ByAppointmentOnly_FALSE	2.4
20	ByAppointmentOnly_TRUE	0.1
21	WheelechairAccessible_(Missing)	10.9
22	WheelechairAccessible_FALSE	0.6
23	WheelechairAccessible_TRUE	8.9
24	OutdoorSeating_(Missing)	1.0
25	OutdoorSeating_FALSE	2.2
26	OutdoorSeating_TRUE	1.9
27	RestaurantsReservations_(Missing)	1.4
28	RestaurantsReservations_FALSE	3.2
29	RestaurantsReservations_TRUE	2.0
30	DogsAllowed_(Missing)	7.7
31	DogsAllowed_FALSE	5.5
32	DogsAllowed_TRUE	1.0
33	Caters_(Missing)	1.3
34	Caters_FALSE	2.8
35	Caters_TRUE	3.3

[47]:
```
plt.figure(figsize=(8,7))
plt.barh(X_train_dtc.columns, 100*rf_cv.best_estimator_.feature_importances_)
plt.show()
```



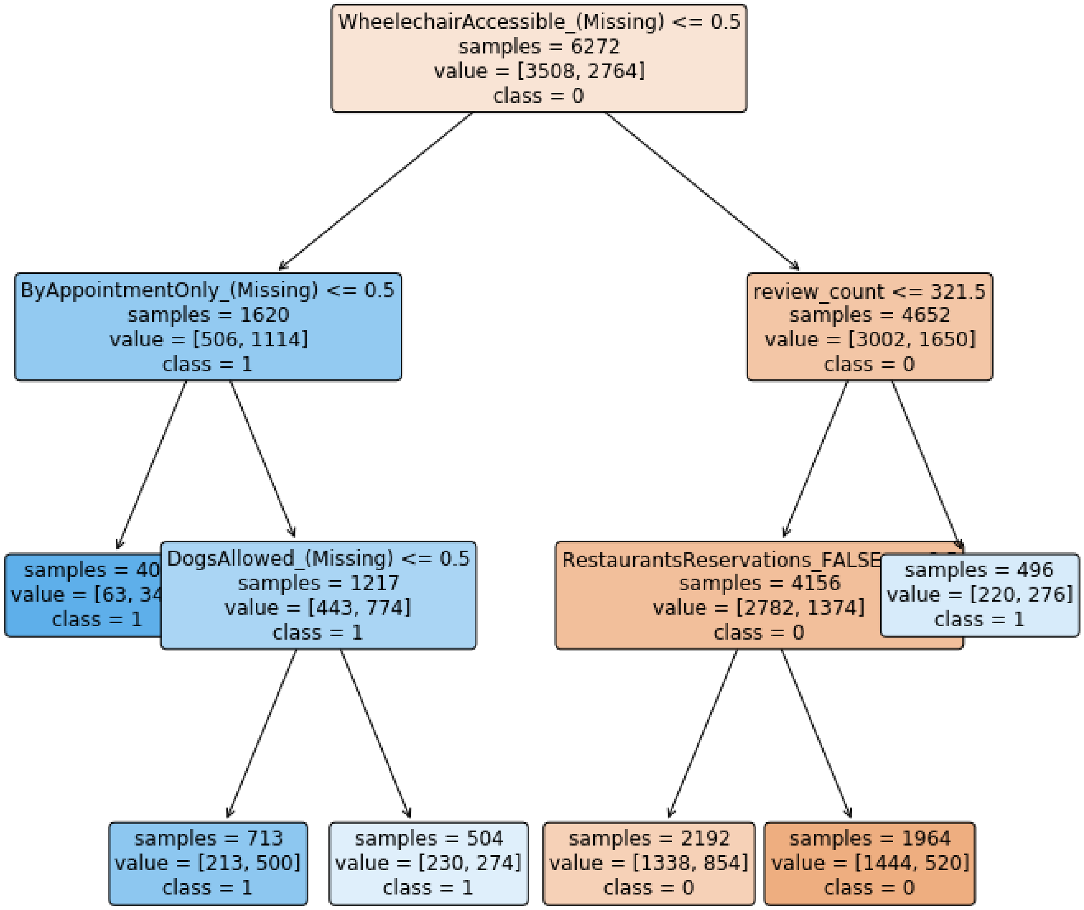
[48]:
```
print(np.
↪average(yelp_train_new[yelp_train_new['fourOrAbove']==1]['review_count']))
print(np.
↪average(yelp_train_new[yelp_train_new['fourOrAbove']==0]['review_count']))
```

```
240.8972503617945
118.69127708095782
```

[49]:
```
from sklearn.tree import plot_tree

print('Node count =', dtc_cv.best_estimator_.tree_.node_count)
plt.figure(figsize=(12,12))
plot_tree(dtc_cv.best_estimator_,
          feature_names=X_train_dtc.columns,
          class_names=['0','1'],
          filled=True,
          impurity=False,
          rounded=True,
          fontsize=12)
plt.show()
```

```
Node count = 11
```



[50]:
```
np.sum(X_train_dtc['WheelechairAccessible_(Missing)']==0)
```

[50]: 1620

4.6 Problem (e) (20 points)

This is an open-ended question.

We used random forest with cv model and CART(Decision Tree Classifier) model to make a recommendation. We are using both models as interpreting the random forest is much more difficult than CART model. The CART model has moderate performance in our case.

First of all, if we take a look at feature importance of our random forest model, our model says that 'review_count' is the most important variable that affects our prediction. The average of review counts of restaurants which have stars great than equal to 4 is 240 and the average of review counts of restaurants which have starts less than 4 is 118. From this data, we can say that bigger review counts helps the restaurant to gain stars great than equal to 4. We could explain to restaurants owner that our model shows the trend that if your restaurant has review count close to 250 than it has higher probability to get stars higher than 4. So it would be preferable to launch a promotion that gives customers an extra side dishes or drinks if they leave any reviews in the Yelp app.

Secondly, let's take a look at the plot of our CART model. If you first take a look at the root node, we can see that missing data of 'Wheelchair accesscibility' would be lead the restaurant to be classified as stars less than 4. So it is important to tell the restaurants owner that there is a high probability that cusomters might not prefer the restaurants which are missing the wheelechair accesscibility of the restaurants. So we should remind each restaurant owners to give the information of wheelechair accessibility if they haven't do it yet.

Finally, let's take a look at the left and right tree of the root node. As we can see, 'review_count' and 'By Appointment Only' are the two important variables. We can see that missing data of 'By Appointment Only' is critical to the restaurant's reputatoin. Also, the importance of the 'review_count' which we checked at the first place is again revisited by the CART model. So similar to the second tip, we should remind each restaurant owners to provide the information of 'By appointment Only' tab since it is critical to their reputation.

4.6.1 Grading Rubrics

- (-3) For each tips that are suggested without using data. Can be deducted upto 9 points.
- (-2) For each tips that are not expressed in a easy language. You should explain to restaurant owners without using any jargons of machine learning. Can be deducted upto 6 points.
- (-5) Nothing done with this question

[ ]:
```

```

[ ]:
```

```