DS5220 Homework 5 (Optional submission)

Aditya Singh

TOTAL POINTS

100 / 100

QUESTION 1

Problem 1 20 pts

1.1 5 / 5

✓ **- 0 pts Correct**

- 5 pts Not done

- 2 pts Incomplete / Partially incorrect solution

1.2 5 / 5

✓ **- 0 pts Correct**

- 5 pts Not Done

- 2 pts Incomplete / Partially incorrect solution

1.3 5 / 5

✓ **- 0 pts Correct**

- 5 pts Not Done,

- 2 pts Incomplete or partially correct solution

1.4 5 / 5

✓ **- 0 pts Correct**

- 5 pts Not Done

- 2 pts Incomplete / Partially incorrect solution

QUESTION 2

Problem 2 40 pts

2.1 0 / 0

✓ **- 0 pts Correct**

2.2 10 / 10

✓ **- 0 pts Correct**

- 10 pts Not Done Q2 (b): Fit a Boosted Regression Tree (BRT) model to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

- 5 pts Incorrect/incomplete plot of the tree and interpretation of the results.

- 2 pts Test MSE not reported

2.3 15 / 15

✓ **- 0 pts Correct**

- 15 pts Not Done Q2.3: Determine the optimal set of parameters for n_estimators, max_depth and min_samples_split for the boosted tree model using cross-validation.

- 5 pts partially incorrect/incomplete solution.

2.4 15 / 15

✓ **- 0 pts Correct**

- 15 pts Not done Q2.4: Based on the selected parameters, make a horizontal bar plot for the feature importance scores and discuss your results.

- 5 pts partially incorrect/incomplete solution.

QUESTION 3

Problem 3 40 pts

3.1 10 / 10

✓ **- 0 pts Correct**

3.2 15 / 15

✓ **- 0 pts Correct**

3.3 15 / 15

✓ **- 0 pts Correct**

- 1 pts unclear explanation as to why it is 1/n

1.1 5 / 5

✓ **- 0 pts** **Correct**

- 5 pts Not done

- 2 pts Incomplete / Partially incorrect solution

1.2 5 / 5

✓ **- 0 pts** **Correct**

- 5 pts Not Done

- 2 pts Incomplete / Partially incorrect solution

1.3 5 / 5

✓ **- 0 pts** **Correct**

- 5 pts Not Done,

- 2 pts Incomplete or partially correct solution

1.4 5 / 5

✓ **- 0 pts** **Correct**

- 5 pts Not Done

- 2 pts Incomplete / Partially incorrect solution

2.1 0 / 0

✓ - **0 pts** Correct

2.2 10 / 10

✓ **- 0 pts Correct**

- 10 pts Not Done Q2 (b): Fit a Boosted Regression Tree (BRT) model to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

- 5 pts Incorrect/incomplete plot of the tree and interpretation of the results.

- 2 pts Test MSE not reported

2.3 15 / 15

✓ **- 0 pts Correct**

- 15 pts Not Done Q2.3: Determine the optimal set of parameters for n_estimators, max_depth and min_samples_split for the boosted tree model using cross-validation.

- 5 pts partially incorrect/incomplete solution.

2.4 15 / 15

✓ **- 0 pts** **Correct**

- 15 pts Not done Q2.4: Based on the selected parameters, make a horizontal bar plot for the feature importance scores and discuss your results.

- 5 pts partially incorrect/incomplete solution.

3.1 10 / 10

✓ - **0 pts** Correct

3.2 15 / 15

✓ - **0 pts** Correct

3.3 15 / 15

✓ **- 0 pts** **Correct**

- 1 pts unclear explanation as to why it is 1/n

Page No.
Date :

Problem - 1.

1-1 In probability distribution, the weighted average of all possible values of a random variable, with weights given by respective theoretical probabilities, is known as the expected value. usually represented by $E(x)$

Since, 'D' is d dimensional it can be written as

$$D = [D_1 \; D_2 \; D_3 \; \cdots \; D_d]^T$$

The expectation of D is

$$E\vec{D} = \begin{bmatrix} E\,D_1 \\ E\,D_2 \\ \vdots \\ E\,D_d \end{bmatrix}$$

covariance matrix :- for a given random vector $D : \Omega \to R^d$ the covariance matrix is $d \times d$ square matrix

$$\Sigma_{ij} = cov[D_i, D_j]$$

$$\Sigma = \begin{bmatrix} cov[D_1, D_1] & \cdots & cov[D_i, D_d] \\ \vdots & \ddots & \vdots \\ cov[D_d, D_1] & \cdots & cov[D_d, D_d] \end{bmatrix}$$

Page No.
Date

$$= \begin{bmatrix} E[D_1^2] - E[D_1]E[D_1] & \cdots & E[D_1 D_d] - E[D_1] \cdot E[D_d] \\ \vdots & & \vdots \\ E[D_d D_1] - E[D_d]E[D_1] & \cdots & E[D_d^2] - E[D_d] \cdot E[D_d] \end{bmatrix}$$

$$= E[DD^T] - E[D]E[D]^T$$

$$= E\left[(D - E[D]) - (D - E[D])^T\right]$$

1.2 A SVD of m x n matrix 'M' is given by the formula.

$$M = U W V^T$$

U: m x n matrix of the orthonormal eigenvectors of $M \cdot M^T$

V^T: transpose of n x n matrix containing the orthonormal egien vectors of $M^T \cdot M$

W :- n x n diagonal matrix of the singular values which are the square root of eigen values of $M^T \cdot M$.

$$C_{m \times n} = U_{m \times r} \times \Sigma_{r \times r} \times V^T_{r \times n}$$

Page No.
Date :

Use cases of SVD.

① Matrix completion

For most recommendation algorithms, the input matrix being very sparse, matrix factorization methods are key since the space needs to be 'reduced' to a smaller latent one. SVD plays a central role in it

② SVD used in PCA

In principal component analysis (PCA) Given a input matrix X. it consists in finding components p_i that are linear combination of original coordinates.

$$p_i = \sum_{j=1}^{d} w_{ij} x_j \quad \text{for } i = 1, \ldots d.$$

while computing the PCA we need eigen vectors of variance-covariance matrix which are calculated by SVD the 'U' gives those values

∴ we can use SVD to perform PCA In some case using SVD to perform PCA is numerically more efficient

Page No.
Date

1.3

Ridge Regression

Ridge regression works with enhanced cost function when compared to the least squares cost function.

$$J(w) = \frac{1}{2N} \sum_{i=1}^{N} \left[\left(w_0 + w_1 x_1^{(i)} \right) - y^{(i)} \right]^2 + \frac{\lambda}{2N} \sum_{j=1}^{2} w_j^2$$

In this expression we can see an additional 'regularization' parameter.

here λ is the hyperparameter we have to hypertune, larger the value of λ the lesser would be slope after gradient descent. Cross validation is one of the ways we can find ideal value for lambda.

Lasso Regression

Lasso Regression works with the following loss function

$$J(w) = \frac{1}{2N} \sum_{i=1}^{N} \left[\left(w_0 + w_1 x_1^{(i)} \right) - y^{(i)} \right]^2 + \frac{\lambda}{2N} \sum_{j=1}^{2} |w_j|$$

In case of lasso instead of taking slope^2 times λ we take absolute value of slopes

In case of lasso Regression we can choose value of λ by hypertunning. Cross-validation is one of the ways we can use λ.

Common features between Ridge and Lasso

Similar to the lasso regularization, ridge puts a similar constraint on the coefficients by introducing a penalty factor. However, while lasso regularization takes magnitude of coefficients, ridge takes the square

Difference between Ridge and Lasso

① When we have a lot of parameters and all those have good correlation with the target variable then we should chose the 'Ridge Regression' and if there are a lot of parameters and many of them are redundant then we should use 'Lasso Regression' as it will bring the slope of those parameter to 0.

1.4 Dropout Regularization.

Dropout regularization means randomly ignoring certain nodes in a layer during training.

Each time the gradient of our model is updated we generate a new thinned neural network based on the different units dropped. given by a probability hyperparameter 'p'

Dropout will ensure that:

1. The neuron's can't rely on one input because it might be dropped out at random. This ~~Cout~~ reduces bias due to over-relying on one input; bias is as major cause of overfitting
2. Neurons will not learn redundant details of inputs This ensures only important information is stored by the neurons This enables the neural network to gain useful knowledge which it uses to make prediction

PS5-2

December 10, 2022

1 Part 2

2-A

[ ]:
```
# imports

import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import plot_tree
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set()
```

[ ]:
```
# Readinf the data
df = pd.read_csv('College.csv')
```

[ ]:
```
#creating X and y for train test split

df.rename(columns={'Unnamed: 0': 'Colleges'},inplace = True)
df['accept_rate'] = df['Accept']/df['Apps']
df = pd.get_dummies(df, columns=['Private'], drop_first=True)
df.drop(columns = ['Accept','Apps'],inplace= True)
X = df.drop(columns = ['accept_rate','Colleges'])
y = df['accept_rate']
```

[ ]:
```
# train test split of the data

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2,␣
 ↪random_state=1)
```

2-B

[ ]:
```
# fitting the gradient boosting model
gbr = GradientBoostingRegressor()
gbr.fit(X_train,y_train)
```

[ ]:
```
GradientBoostingRegressor()
```

[ ]:
```
y_pred = gbr.predict(X_test)
mse = mean_squared_error(y_true = y_test, y_pred= y_pred)
print("mean squared error for the Gradient Boosting Tree is {0:3f}".format(mse))
```

```
mean squared error for the Gradient Boosting Tree is 0.013595
```

[ ]:
```
gbr.feature_importances_
```

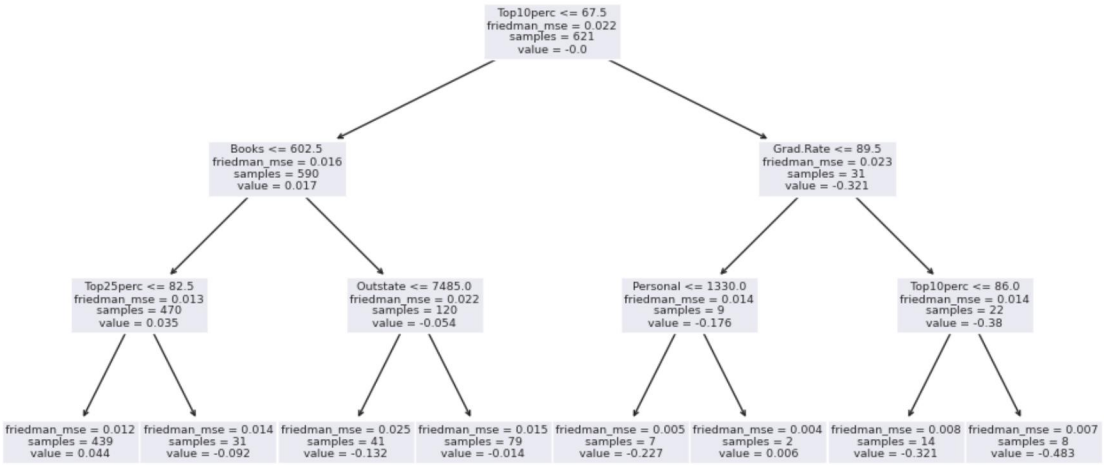
[ ]:
```
array([0.01073169, 0.24859828, 0.11355859, 0.02834218, 0.03592775,
       0.14923082, 0.07837342, 0.06856991, 0.03567802, 0.01066301,
       0.01135948, 0.03686528, 0.01911258, 0.0187745 , 0.13421448,
       0.        ])
```

[ ]:
```
pd.DataFrame(columns = ["Features","Coefficient"] , data = (zip(X.columns,gbr.
  ↪feature_importances_)))
```

[ ]:

	Features	Coefficient
0	Enroll	0.011194
1	Top10perc	0.248198
2	Top25perc	0.113467
3	F.Undergrad	0.028207
4	P.Undergrad	0.035512
5	Outstate	0.149328
6	Room.Board	0.080376
7	Books	0.068981
8	Personal	0.033619
9	PhD	0.015259
10	Terminal	0.006445
11	S.F.Ratio	0.036949
12	perc.alumni	0.018794
13	Expend	0.019545
14	Grad.Rate	0.134126
15	Private_Yes	0.000000

[ ]:
```
plt.figure(figsize=(12, 6))
plot_tree(gbr.estimators_[0][0], max_depth=5, feature_names=X.columns)
plt.show()
```



As we can see above the Gradient Boosting Tree for better visualization I have chosen max_dept=5, with mean squared error being 0.013595

———————————————————————————2-C———————————————————————————-

[ ]:
```
#parameters list
tree_param = {'max_depth' : [ i for i in range(1,11)],
              'n_estimators': [15, 20, 50, 75, 100, 150, 200, 400],
               'min_samples_split' : [ i for i in range(1,11)]
              }
```

[ ]:
```
#Grid Search CV for getting optimal parameters
gbr_cv = GridSearchCV(GradientBoostingRegressor(), tree_param, cv=5,verbose=1)
```

[ ]:
```
gbr_cv.fit(X_train,y_train)
```

[ ]:
```
pd.DataFrame(gbr_cv.best_params_,index=["Best Parameters"])
```

[ ]:

	max_depth	min_samples_split	n_estimators
Best Parameters	2	4	75

Based on The Cross Validation results ‘max_depth’ = 2, ‘min_sample_split = 4’ and ‘n_estimators = 7’ are the optimal values

———————————————————————————-2-D———————————————————————————

[ ]:
```
# Fitting Gradient Boosting Tree based on the Parameters found through Cross␣
↪Validation
```

```
gbr_cv = GradientBoostingRegressor(max_depth = 2, min_samples_split =4,␣
 ↪n_estimators= 75)
gbr_cv.fit(X_train,y_train)
```

[ ]:
```
GradientBoostingRegressor(max_depth=2, min_samples_split=4, n_estimators=75)
```

[ ]:
```
y_pred = gbr_cv.predict(X_test)
mse = mean_squared_error(y_true = y_test, y_pred= y_pred)
print("mean squared error for the CV Gradient Boosting Tree is {0:3f}".
 ↪format(mse))
```

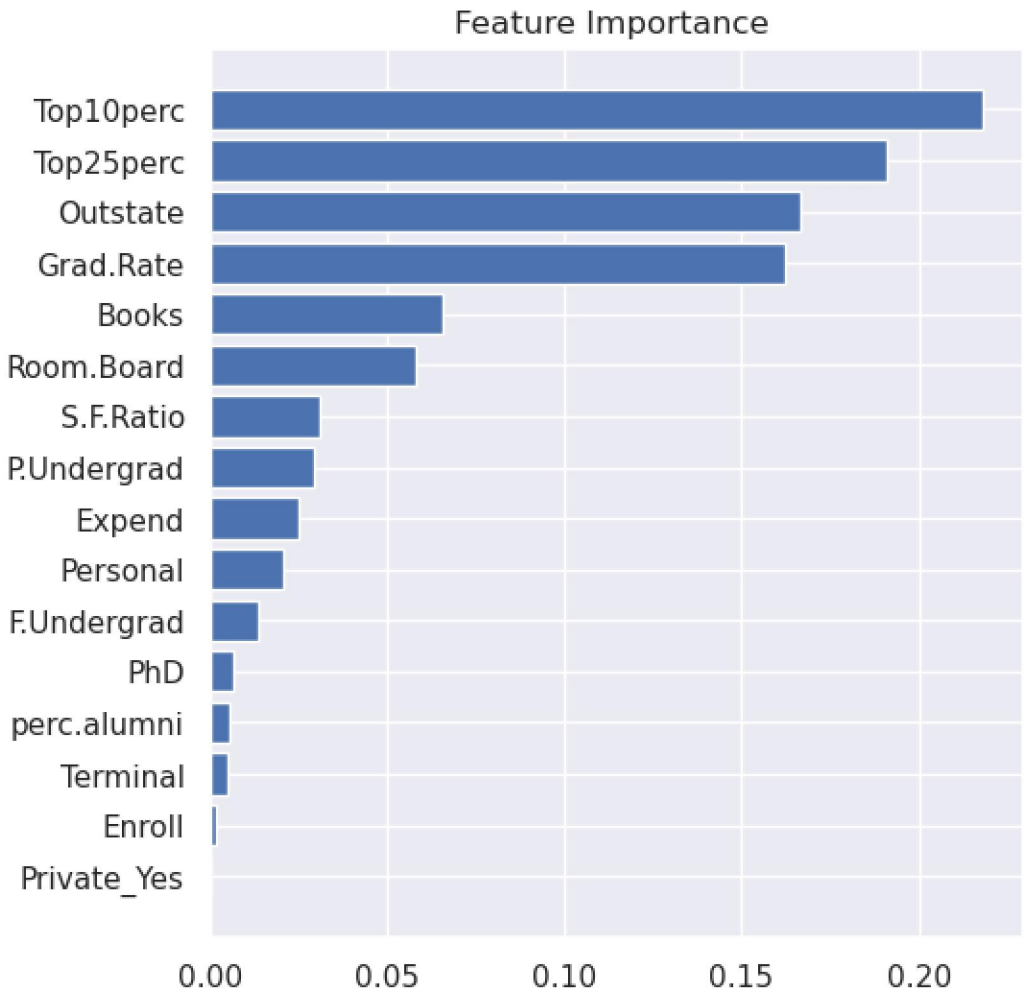
```
mean squared error for the CV Gradient Boosting Tree is 0.012238
```

As we can see that the MSE loss has reduced after performing th cross validation form 0.013595 to 0.0122238

[ ]:
```
# plotting the Feature importance Descending as per their importance

feature_importance = gbr_cv.feature_importances_
sorted_idx = np.argsort(feature_importance)
pos = np.arange(sorted_idx.shape[0]) + 0.5
fig = plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.barh(pos, feature_importance[sorted_idx], align="center")
plt.yticks(pos, np.array(X.columns)[sorted_idx])
plt.title("Feature Importance ")
```

[ ]:
```
Text(0.5, 1.0, 'Feature Importance ')
```


Feature Importance
Top10perc
Top25perc
Outstate
Grad.Rate
Books
Room.Board
S.F.Ratio
P.Undergrad
Expend
Personal
F.Undergrad
PhD
perc.alumni
Terminal
Enroll
Private_Yes
0.00
0.05
0.10
0.15
0.20

Page No.
Date :

Problem - 3

3.1 $P_j = \frac{\exp(z_j)}{\sum_{i=1}^{k} \exp(z_i)}$ for any $j = 1, 2, 3 \ldots k$

$$\therefore \frac{\partial P_j}{\partial z_k} = \frac{\partial}{\partial z_k}\left\{\frac{\exp(z_j)}{\sum_{i=1}^{k} \exp(z_i)}\right\}$$

Now we will apply the U/V rule

$$\frac{\partial P_j}{\partial z_k} = \frac{\left(\frac{\partial}{\partial z_k} \exp(z_j)\right) * \Sigma - \exp(z_j) \frac{\partial}{\partial z_k} \Sigma}{\Sigma^2}$$

Now in case of $k = j$ we will have

$$\frac{\partial}{\partial z_k} \exp(z_j) = \exp(z_j) \quad \text{—①}$$

And for $k \neq j$, $\frac{\partial}{\partial z_k} \exp(z_j) = 0$ —②

Also,

$$\frac{\partial}{\partial z_k} \sum_{i=1}^{k} \exp(z_i) = \exp(z_k) \text{ irrespective of } k = j \text{ or } k \neq j \quad \text{—③}$$

∴ Case ① $k = j$, from eq^n ① & ③

$$\frac{\partial P_j}{\partial z_k} = \frac{\exp(z_j) * \Sigma - \exp(z_j) * \exp(z_k)}{\Sigma^2}$$

$$\frac{\partial P_j}{\partial z_k} = \frac{\exp(z_j)}{\Sigma}\left[\frac{\Sigma}{\Sigma} - \frac{\exp(z_k)}{\Sigma}\right]$$

Page No.
Date :

$$\therefore \frac{\partial P_i}{\partial z_k} = S_j (1 - S_k) \quad \text{---} \quad (4)$$
$$\frac{\partial P_i}{\partial z_k} = S_j (1 - S_j)$$

where $S_j \rightarrow$ softmax function applied on 'j'
$S_k \rightarrow$ softmax function applied on 'k'

Case 2, $k \neq j$ from eq (2), (3)

$$\frac{\partial}{\partial z_k} P_j = \frac{0 * \Sigma - \exp(z_j) * \exp(z_k)}{\Sigma^2}$$

$$= \frac{-\exp(z_j)}{\Sigma} * \frac{\exp(z_k)}{\Sigma}$$

$$= -S_j * S_k \quad \ldots \text{ from am}$$

$$\therefore \frac{\partial}{\partial z_k} P_j = \begin{cases} S_j(1 - S_k) & k = j \\ -S_j * S_k & k \neq j \end{cases}$$

Page No.
Date :

3.2 The results we got from 3.1

$$\frac{\partial P_j}{\partial z_k} = \begin{cases} S_j(1 - S_k) & k = j \\ -S_j * S_k & k \neq j \end{cases}$$

For the case $k = j$

$$\frac{\partial}{\partial z_k} P_j = S_j (1 - S_k)$$

$$= \frac{\exp(z_j)}{\sum_{i=1}^{K} \exp(z_i)} \left[1 - \frac{\exp(z_k)}{\sum_{i=1}^{K} \exp(z_i)} \right]$$

from the above equation we can see that as we increase the value of z_j the corresponding change in probability will also increase positively.

For the case of $k \neq j$

$$\frac{\partial P_j}{\partial z_k} = -S_j * S_k$$

$$= -\frac{\exp(z_j)}{\sum_{i=1}^{K} \exp(z_i)} * \frac{\exp(z_k)}{\sum_{i=1}^{K} \exp(z_i)}$$

from the above equation we can see that as we increase the value of 'z_j' the corresponding change in 'P_j' is negative therefore the value decreases with increase in 'z_j' for $k \neq j$

Page No.
Date :

3.3 $q_j(c) = \dfrac{\exp(c \cdot z_j)}{\sum_{i=1}^{K} \exp(c \cdot z_i)}$ for any $j = 1, 2, 3 \cdots K$

$M = \max_{i=1}^{K} z_i$, be the maximum over all z_i for $i = 1, 2, 3 \cdots K$

there are 'n' entries whose value is 'M'

$$\lim_{c \to \infty} q_j(c) = \lim_{c \to \infty} \frac{\exp(c \cdot z_j)}{\sum_{i=1}^{K} \exp(c \cdot z_i)}$$

Now assuming $n < K$ i.e not all values are same

$$\lim_{c \to \infty} q_j(c) = \lim_{c \to \infty} \left[\frac{\exp(c \cdot z_j)}{\sum_{i=1}^{K} \exp(c \cdot z_i)} \right]$$

$$= \lim_{c \to \infty} \left[\frac{\exp(c \cdot z_j)}{n \cdot \exp(c \cdot M) + \sum_{i=1}^{K-n} \exp(c \cdot z_i)} \right]$$

dividing Neumerator and denominator by $\exp(c \cdot z_j)$

$$\lim_{c \to \infty} q_j(c) = \lim_{c \to \infty} \frac{1}{\frac{n \cdot \exp(c \cdot M)}{\exp(c \cdot z_j)} + \sum_{i=1}^{K-n} \frac{\exp(c \cdot z_i)}{\exp(c \cdot z_j)}}$$

Page No.
Date :

Case (1) $z_j < M$.

$$\therefore \text{LHS} = \lim_{c \to \infty} \frac{1}{n \cdot \exp(c \cdot (M - z_j)) + \sum_{i=1}^{k-n} \exp(c \cdot (z_i - z_j))} \quad \text{— (1)}$$

put $c = \infty$, we get,

$$\frac{1}{n \cdot \exp(\infty (M - z_j)) + \sum_{i=1}^{k-n} \exp(\infty)}$$

$e^{\infty} = \infty$ and $\frac{1}{\infty} = 0$

$$\therefore \lim_{c \to \infty} q_j(c) = 0 \quad \text{..... (2)}$$

Case (2) :- $z_j = M$. $\therefore$ from eq (1)

$$\text{LHS} = \frac{1}{n \cdot \exp(0) + \sum_{i=1}^{k-n} \exp(c \cdot (z_i - z_j))}$$

put $c = \infty$

$$= \frac{1}{n \cdot (1) + \sum_{i=1}^{k-n} \exp(-\infty)}$$

$$= \quad \because z_j = M, \; z_i - z_j = -ve$$

Page No.
Date :

$$\therefore \lim_{C \to \infty} q_j(C) = \frac{1}{n} \quad //$$

$$\therefore \lim_{C \to \infty} q_j(C) = \begin{cases} 0 & \text{if } z_j < M \\ 1/n & z_j = M \end{cases}$$