Name: Ashish Verma   Course:CS773   HW#5

Solution1:

Given

A	P	L	I	K	Ec	C	Ed	Class
31823	663	223	182	1.23	0.58	32274	201	SEKER
27275	605	220	158	1.39	0.69	27604	186	DERMASON
32799	654	220	190	1.16	0.5	33087	204	SEKER
58434	981	396	190	2.09	0.88	59309	273	HOROZ
68513	1015	359	244	1.47	0.73	69406	295	BARBUNYA
85702	1107	428	257	1.66	0.8	86542	330	CALI
137358	1365	508	345	1.47	0.73	138093	418	BOMBAY
41643	769	295	181	1.63	0.79	42233	230	SIRA
68551	1025	356	246	1.45	0.72	69684	295	BARBUNYA
137115	1427	519	337	1.54	0.76	138970	418	BOMBAY
27277	605	218	159	1.37	0.68	27611	186	DERMASON
41646	762	286	186	1.53	0.76	42074	230	SIRA
85666	1119	436	251	1.73	0.82	86305	330	CALI
58454	965	392	196	2	0.87	60280	273	HOROZ
58484	956	382	197	1.94	0.86	59456	273	HOROZ
41646	768	288	186	1.55	0.76	42225	230	SIRA
27267	597	215	162	1.33	0.66	27575	186	DERMASON

Python Implementation

```python
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import KBinsDiscretizer

# Create the DataFrame
data = {
    'A': [31823, 27275, 32799, 58434, 68513, 85702, 137358, 41643, 68551, 137115, 27277, 41646, 85666, 58454, 58484, 41646, 27267],
    'P': [663, 605, 654, 981, 1015, 1107, 1365, 769, 1025, 1427, 605, 762, 1119, 965, 956, 768, 597],
    'L': [223, 220, 220, 396, 359, 428, 508, 295, 356, 519, 218, 286, 436, 392, 382, 288, 215],
    'I': [182, 158, 190, 190, 244, 257, 345, 181, 246, 337, 159, 186, 251, 196, 197, 186, 162],
    'K': [1.23, 1.39, 1.16, 2.09, 1.47, 1.66, 1.47, 1.63, 1.45, 1.54, 1.37, 1.53, 1.73, 2.00, 1.94, 1.55, 1.33],
    'Ec': [0.58, 0.69, 0.5, 0.88, 0.73, 0.8, 0.73, 0.79, 0.72, 0.76, 0.68, 0.76, 0.82, 0.87, 0.86, 0.76, 0.66],
    'C': [32274, 27604, 33087, 59309, 69406, 86542, 138093, 42233, 69684, 138970, 27611, 42074, 86305, 60280, 59456, 42225, 27575],
    'Ed': [201, 186, 204, 273, 295, 330, 418, 230, 295, 418, 186, 230, 330, 273, 273, 230, 186],
    'Class': ['SEKER', 'DERMASON', 'SEKER', 'HOROZ', 'BARBUNYA', 'CALI', 'BOMBAY', 'SIRA', 'BARBUNYA', 'BOMBAY', 'DERMASON', 'SIRA', 'CALI', 'HOROZ', 'HOROZ', 'SIRA', 'DERMASON']
}

df = pd.DataFrame(data)

# (i) Equal-width binning (4 bins)
equal_width_bins = KBinsDiscretizer(n_bins=4, encode='ordinal', strategy='uniform')
df['K_equal_width'] = equal_width_bins.fit_transform(df[['K']]) + 1

# (ii) Equal frequency binning (4 bins)
equal_freq_bins = KBinsDiscretizer(n_bins=4, encode='ordinal', strategy='quantile')
df['K_equal_freq'] = equal_freq_bins.fit_transform(df[['K']]) + 1

# (iii) Entropy-based discretization
X = df[['K']].values
y = df['Class'].values

# Use DecisionTreeClassifier for entropy-based discretization
tree = DecisionTreeClassifier(criterion='entropy', max_leaf_nodes=4)
tree.fit(X, y)
df['K_entropy'] = tree.apply(X)

# Sort the DataFrame by K
df_sorted = df.sort_values(by='K').reset_index(drop=True)

# Add 1 to K_entropy bins for consistency
df_sorted['K_entropy'] = df_sorted['K_entropy'] - df_sorted['K_entropy'].min() + 1

# Display the sorted results
print(df_sorted[['K', 'K_equal_width', 'K_equal_freq', 'K_entropy']])
```

```
       K  K_equal_width  K_equal_freq  K_entropy
0   1.16            1.0           1.0          3
1   1.23            1.0           1.0          3
2   1.33            1.0           1.0          3
3   1.37            1.0           1.0          3
4   1.39            1.0           2.0          3
5   1.45            2.0           2.0          4
6   1.47            2.0           2.0          4
7   1.47            2.0           2.0          4
8   1.53            2.0           3.0          1
9   1.54            2.0           3.0          1
10  1.55            2.0           3.0          1
11  1.63            3.0           3.0          1
12  1.66            3.0           4.0          2
13  1.73            3.0           4.0          2
14  1.94            4.0           4.0          2
15  2.00            4.0           4.0          2
16  2.09            4.0           4.0          2
```

Manual Calculations

Equal-width binning:

Range of data

Max value of K:-2.09

Min value of K:-1.16

Calculate the range

Range = Max – Min = 2.09 – 1.16 = 0.93

Calculate bin width

Bin width = Range/number of bins = 0.93/4= 0.2325

Determine Bin Edges

- Bin 1: [1.16, 1.16 + 0.2325) = [1.16, 1.3925)
- Bin 2: [1.3925, 1.3925 + 0.2325) = [1.3925, 1.625)
- Bin 3: [1.625, 1.625 + 0.2325) = [1.625, 1.8575)
- Bin 4: [1.8575, 1.8575 + 0.2325) = [1.8575, 2.09)

Assign values of K

- K=1.23 falls into Bin 1: [1.16, 1.3925)
- K=1.39 falls into Bin 1: [1.16, 1.3925)
- K=1.16 falls into Bin 1: [1.16, 1.3925)
- K=2.09 falls into Bin 4: [1.8575, 2.09)
- K=1.47 falls into Bin 2: [1.3925, 1.625)
- K=1.66 falls into Bin 3: [1.625, 1.8575)
- K=1.47 falls into Bin 2: [1.3925, 1.625)
- K=1.63 falls into Bin 3: [1.625, 1.8575)
- K=1.45 falls into Bin 2: [1.3925, 1.625)
- K=1.54 falls into Bin 2: [1.3925, 1.625)
- K=1.37 falls into Bin 1: [1.16, 1.3925)
- K=1.53 falls into Bin 2: [1.3925, 1.625)
- K=1.73 falls into Bin 3: [1.625, 1.8575)
- K=2.00 falls into Bin 4: [1.8575, 2.09)
- K=1.94 falls into Bin 4: [1.8575, 2.09)
- K=1.55 falls into Bin 2: [1.3925, 1.625)
- K=1.33 falls into Bin 1: [1.16, 1.3925)

Equal frequency binning:

Sort the data in ascending order:

- Sorted K values: [1.16, 1.23, 1.33, 1.37, 1.39, 1.45, 1.47, 1.47, 1.53, 1.54, 1.55, 1.63, 1.66, 1.73, 1.94, 2.00, 2.09]

Determine the number of elements per bin

- Total number of elements: 17
- Number of bins: 4
- Elements per bin: 17/4=4.25
- Since we can't have a fraction of an element, bins will have either 4 or 5 elements.

Assign each value to a bin

- Bin 1: [1.16, 1.23, 1.33, 1.37, 1.39] (first 5 elements)
- Bin 2: [1.45, 1.47, 1.47, 1.53, 1.54] (next 5 elements)
- Bin 3: [1.55, 1.63, 1.66, 1.73] (next 4 elements)
- Bin 4: [1.94, 2.00, 2.09] (last 3 elements)

K	K_Binned_EF
1.16	1
1.23	1
1.33	1
1.37	1
1.39	2
1.45	2
1.47	2
1.47	2
1.53	3
1.54	3
1.55	3
1.63	3
1.66	4
1.73	4
1.94	4
2	4
2.09	4

Entropy-based discretization:

Sorted K values:

[1.16,1.33,1.37,1.39,1.45,1.47,1.47,1.53,1.54,1.63,1.66,1.73,1.94,2.00,2.09]

Let's calculate the entropy for a specific cut point, K=1.63

Let’s Split the Data(Split 1)

Left Split (K ≤ 1.63):

[1.16,1.33,1.37,1.39,1.45,1.47,1.47,1.53,1.54,1.63]

Right Split (K > 1.63):

[1.66,1.73,1.94,2.00,2.09]

Calculate the entropy for each split

P(SEKER)=2/11 ,P(DERMASON)=3/11, P(BARBUNYA)=2/11, P(BOMBAY)=2/11, P(SIRA)=3/11

Entropy(Left) = -(2/11*log_2(2/11) + 3/11*log_2 (3/11) +2/11*log_2(2/11) +2/11*log_2(2/11) + 3/11*log_2(3/11)

= -(-2.636)

= 2.636

P(CALI)=2/5, P(HOROZ)=3/5

Entropy(Right) = -(2/6 * log_2(2/6) + 3/6*log_2(3/6))

=1.028

Weighted Entropy = 10/17*2.636+ 5/17 *1.028

= 1.8529

Bin1 : [1.16, 1.63]

Bin2 : [1.63, 2.09]

Let’s Split the Data(Split 2)

Left Split (K ≤ 1.53):

[1.16,1.33,1.37,1.39,1.45,1.47,1.47,1.53]

Right Split (K > 1.53):

[1.54,1.63,1.66,1.73,1.94,2.00,2.09]

Calculate the entropy for each split

P(SEKER)=2/9 ,P(DERMASON)=3/9, P(BARBUNYA)=2/9, P(BOMBAY)=1/9, P(SIRA)=1/9

Entropy(Left) = −(2/9*log_2(2/9) + 3/9*log_2(3/9) +2/9*log_2(2/9) +1/9*log_2(1/9) + 1/9*log_2(1/9))

= -(-1.9056)

= 1.9056

P(BOMBAY)=1/8, P(SIRA)=2/8 , P(CALI)=2/8, P(HOROZ)=3/8

Entropy(Right)= -(1/8*log_2(1/8) + 2/8*log_2(2/8) + 2/8*log_2(2/8) +3/8*log_2(3/8))

=-(-1.9056)

=1.9056

Weighted Entropy = 9/17*1.9056+ 8/17 *1.9056

= 1.9056

Bin1 : [1.16, 1.53]

Bin2 : [1.53, 2.09]

Since weighted entropy for Split1 is lower than split2

Bin1 : [1.16, 1.63]

Bin2 : [1.63, 2.09]

Is better binning

Since there could be various cut points, we need to find the best cut which have lowest weighted entropy.

K_entropy column in above table shows the actual binning based on entropy the minimum value of K split is coming as 1.47 as shown I below table

K	Information Gain
1.16	0.132
1.23	0.45
1.33	0.437
1.37	0.552
1.39	0.801
1.45	0.746
1.47	**0.807**
1.53	0.645
1.54	0.734
1.55	0.702
1.63	0.801
1.66	0.597
1.73	0.6
1.94	0.288
2	0.088
2.09	0.072

Hence, we can split the bin as

Bin1 [1.16,1.47]

Bin2 [1.47,2.09]

Solution2:-

Python Implementation

Standard Method

```python
import pandas as pd

# Given data
data = {
    "A": [31823, 27275, 32799, 58434, 68513, 85702, 137358, 41643, 68551, 137115, 27277, 41646, 85666, 58454, 58484, 41646, 27267],
    "P": [663, 605, 654, 981, 1015, 1107, 1365, 769, 1025, 1427, 605, 762, 1119, 965, 956, 768, 597],
    "L": [223, 220, 220, 396, 359, 428, 508, 295, 356, 519, 218, 286, 436, 392, 382, 288, 215],
    "I": [182, 158, 190, 190, 244, 257, 345, 181, 246, 337, 159, 186, 251, 196, 197, 186, 162],
    "K": [1.23, 1.39, 1.16, 2.09, 1.47, 1.66, 1.47, 1.63, 1.45, 1.54, 1.37, 1.53, 1.73, 2.0, 1.94, 1.55, 1.33],
    "Ec": [0.58, 0.69, 0.5, 0.88, 0.73, 0.8, 0.73, 0.79, 0.72, 0.76, 0.68, 0.76, 0.82, 0.87, 0.86, 0.76, 0.66],
    "C": [32274, 27604, 33087, 59309, 69406, 86542, 138093, 42233, 69684, 138970, 27611, 42074, 86305, 60280, 59456, 42225, 27575],
    "Ed": [201, 186, 204, 273, 295, 330, 418, 230, 295, 418, 186, 230, 330, 273, 273, 230, 186],
    "Class": ["SEKER", "DERMASON", "SEKER", "HOROZ", "BARBUNYA", "CALI", "BOMBAY", "SIRA", "BARBUNYA", "BOMBAY", "DERMASON", "SIRA", "CALI", "HOROZ", "HOROZ", "SIRA", "DERMASON"]
}

# Create DataFrame
df = pd.DataFrame(data)

# Applying one-hot encoding
binary_df = pd.get_dummies(df, columns=["Class"], drop_first=False)

print("Transformed DataFrame using one-hot encoding:")
print(binary_df.head())
```

```
Transformed DataFrame using one-hot encoding:
       A     P    L    I     K    Ec       C   Ed  Class_BARBUNYA  \
0  31823   663  223  182  1.23  0.58   32274  201           False
1  27275   605  220  158  1.39  0.69   27604  186           False
2  32799   654  220  190  1.16  0.50   33087  204           False
3  58434   981  396  190  2.09  0.88   59309  273           False
4  68513  1015  359  244  1.47  0.73   69406  295            True

   Class_BOMBAY  Class_CALI  Class_DERMASON  Class_HOROZ  Class_SEKER  \
0         False       False           False        False         True
1         False       False            True        False        False
2         False       False           False        False         True
3         False       False           False         True        False
4         False       False           False        False        False

   Class_SIRA
0       False
1       False
2       False
3       False
4       False
```

Error Correcting Code using Hamming method

```python
import pandas as pd

def hamming_encode(data):
    # Check input data length
    if len(data) != 4:
        raise ValueError("Input data length must be 4 bits.")

    # Calculate parity bits
    p1 = data[0] ^ data[1] ^ data[3]
    p2 = data[0] ^ data[2] ^ data[3]
    p3 = data[1] ^ data[2] ^ data[3]

    # Create codeword
    codeword = [p1, p2, data[0], p3, data[1], data[2], data[3]]

    return codeword

# Original data
data = {
    'A': [31823, 27275, 32799, 58434, 68513, 85702, 137358, 41643, 68551, 137115, 27277, 41646, 85666, 58454, 58484, 41646, 27267],
    'P': [663, 605, 654, 981, 1015, 1107, 1365, 769, 1025, 1427, 605, 762, 1119, 965, 956, 768, 597],
    'L': [223, 220, 220, 396, 359, 428, 508, 295, 356, 519, 218, 286, 436, 392, 382, 288, 215],
    'I': [182, 158, 190, 190, 244, 257, 345, 181, 246, 337, 159, 186, 251, 196, 197, 186, 162],
    'K': [1.23, 1.39, 1.16, 2.09, 1.47, 1.66, 1.47, 1.63, 1.45, 1.54, 1.37, 1.53, 1.73, 2.0, 1.94, 1.55, 1.33],
    'Ec': [0.58, 0.69, 0.5, 0.88, 0.73, 0.8, 0.73, 0.79, 0.72, 0.76, 0.68, 0.76, 0.82, 0.87, 0.86, 0.76, 0.66],
    'Ed': [32274, 27604, 33087, 59309, 69406, 86542, 138093, 42233, 69684, 138970, 27611, 42074, 86305, 60280, 59456, 42225, 27575],
    'Class': ['SEKER', 'DERMASON', 'SEKER', 'HOROZ', 'BARBUNYA', 'CALI', 'BOMBAY', 'SIRA', 'BARBUNYA', 'BOMBAY', 'DERMASON', 'SIRA', 'CALI', 'HOROZ', 'HOROZ', 'SIRA', 'DERMASON']
}
# Create DataFrame
df = pd.DataFrame(data)

# Define encoding for each class
class_encoding = {
    'SEKER': [0, 0, 0, 0],
    'DERMASON': [0, 0, 0, 1],
    'HOROZ': [0, 0, 1, 0],
    'BARBUNYA': [0, 0, 1, 1],
    'CALI': [0, 1, 0, 0],
    'BOMBAY': [0, 1, 0, 1],
    'SIRA': [0, 1, 1, 0]
}
# Apply encoding to create binary variables
for cls, encoding in class_encoding.items():
    df[cls] = df['Class'].apply(lambda x: hamming_encode(encoding) if x == cls else [0, 0, 0, 0])

# Extract bits from codeword
for i in range(7):
    df[f'Bit_{i+1}'] = df.apply(lambda row: row[row['Class']][i], axis=1)
# Drop the original 'Class' column and encoded columns
df.drop(columns=['Class', 'SEKER', 'DERMASON', 'HOROZ', 'BARBUNYA', 'CALI', 'BOMBAY', 'SIRA'], inplace=True)
print(df)
```

```
         A     P    L    I     K    Ec      Ed  Bit_1  Bit_2  Bit_3  Bit_4  \
0    31823   663  223  182  1.23  0.58   32274      0      0      0      0
1    27275   605  220  158  1.39  0.69   27604      1      1      0      1
2    32799   654  220  190  1.16  0.50   33087      0      0      0      0
3    58434   981  396  190  2.09  0.88   59309      0      1      0      1
4    68513  1015  359  244  1.47  0.73   69406      1      0      0      0
5    85702  1107  428  257  1.66  0.80   86542      1      0      0      1
6   137358  1365  508  345  1.47  0.73  138093      0      1      0      0
7    41643   769  295  181  1.63  0.79   42233      1      1      0      0
8    68551  1025  356  246  1.45  0.72   69684      1      0      0      0
9   137115  1427  519  337  1.54  0.76  138970      0      1      0      0
10   27277   605  218  159  1.37  0.68   27611      1      1      0      1
11   41646   762  286  186  1.53  0.76   42074      1      1      0      0
12   85666  1119  436  251  1.73  0.82   86305      1      0      0      1
13   58454   965  392  196  2.00  0.87   60280      0      1      0      1
14   58484   956  382  197  1.94  0.86   59456      0      1      0      1
15   41646   768  288  186  1.55  0.76   42225      1      1      0      0
16   27267   597  215  162  1.33  0.66   27575      1      1      0      1

    Bit_5  Bit_6  Bit_7
0       0      0      0
1       0      0      1
2       0      0      0
3       0      1      0
4       0      1      1
5       1      0      0
6       1      0      1
7       1      1      0
8       0      1      1
9       1      0      1
10      0      0      1
11      1      1      0
12      1      0      0
13      0      1      0
14      0      1      0
15      1      1      0
16      0      0      1
```

Nested dichotomies

```
import pandas as pd

# Original data
data = {
    'A': [31823, 27275, 32799, 58434, 68513, 85702, 137358, 41643, 68551, 137115, 27277, 41646, 85666, 58454, 58484, 41646, 27267],
    'P': [663, 605, 654, 981, 1015, 1107, 1365, 769, 1025, 1427, 605, 762, 1119, 965, 956, 768, 597],
    'L': [223, 220, 220, 396, 359, 428, 508, 295, 356, 519, 218, 286, 436, 392, 382, 288, 215],
    'I': [182, 158, 190, 190, 244, 257, 345, 181, 246, 337, 159, 186, 251, 196, 197, 186, 162],
    'K': [1.23, 1.39, 1.16, 2.09, 1.47, 1.66, 1.47, 1.63, 1.45, 1.54, 1.37, 1.53, 1.73, 2.0, 1.94, 1.55, 1.33],
    'Ec': [0.58, 0.69, 0.5, 0.88, 0.73, 0.8, 0.73, 0.79, 0.72, 0.76, 0.68, 0.76, 0.82, 0.87, 0.86, 0.76, 0.66],
    'Ed': [32274, 27604, 33087, 59309, 69406, 86542, 138093, 42233, 69684, 138970, 27611, 42074, 86305, 60280, 59456, 42225, 27575],
    'Class': ['SEKER', 'DERMASON', 'SEKER', 'HOROZ', 'BARBUNYA', 'CALI', 'BOMBAY', 'SIRA', 'BARBUNYA', 'BOMBAY', 'DERMASON', 'SIRA', 'CALI', 'HOROZ', 'HOROZ', 'SIRA', 'DERMASON']
}

# Create DataFrame
df = pd.DataFrame(data)

# Identify unique classes
unique_classes = df['Class'].unique()

# Create binary variables using nested dichotomies
for i, cls1 in enumerate(unique_classes):
    for cls2 in unique_classes[i+1:]:
        # Create new column with binary values
        df[f"{cls1}_vs_{cls2}"] = (df['Class'] == cls1).astype(int) - (df['Class'] == cls2).astype(int)

# Drop the original 'Class' column
df.drop(columns=['Class'], inplace=True)

print(df)
```

```
         A     P    L    I     K    Ec      Ed  SEKER_vs_DERMASON  \
0    31823   663  223  182  1.23  0.58   32274                  1   
1    27275   605  220  158  1.39  0.69   27604                 -1   
2    32799   654  220  190  1.16  0.50   33087                  1   
3    58434   981  396  190  2.09  0.88   59309                  0   
4    68513  1015  359  244  1.47  0.73   69406                  0   
5    85702  1107  428  257  1.66  0.80   86542                  0   
6   137358  1365  508  345  1.47  0.73  138093                  0   
7    41643   769  295  181  1.63  0.79   42233                  0   
8    68551  1025  356  246  1.45  0.72   69684                  0   
9   137115  1427  519  337  1.54  0.76  138970                  0   
10   27277   605  218  159  1.37  0.68   27611                 -1   
11   41646   762  286  186  1.53  0.76   42074                  0   
12   85666  1119  436  251  1.73  0.82   86305                  0   
13   58454   965  392  196  2.00  0.87   60280                  0   
14   58484   956  382  197  1.94  0.86   59456                  0   
15   41646   768  288  186  1.55  0.76   42225                  0   
16   27267   597  215  162  1.33  0.66   27575                 -1   

    SEKER_vs_HOROZ  SEKER_vs_BARBUNYA  ...  HOROZ_vs_BARBUNYA  HOROZ_vs_CALI  \
0                1                  1  ...                  0              0   
1                0                  0  ...                  0              0   
2                1                  1  ...                  0              0   
3               -1                  0  ...                  1              1   
4                0                 -1  ...                 -1              0   
5                0                  0  ...                  0             -1   
6                0                  0  ...                  0              0   
7                0                  0  ...                  0              0   
8                0                 -1  ...                 -1              0   
9                0                  0  ...                  0              0   
10               0                  0  ...                  0              0   
11               0                  0  ...                  0              0   
12               0                  0  ...                  0             -1   
13              -1                  0  ...                  1              1   
14              -1                  0  ...                  1              1   
15               0                  0  ...                  0              0   
16               0                  0  ...                  0              0   

    HOROZ_vs_BOMBAY  HOROZ_vs_SIRA  BARBUNYA_vs_CALI  BARBUNYA_vs_BOMBAY  \
```

```
    HOROZ_vs_BOMBAY  HOROZ_vs_SIRA  BARBUNYA_vs_CALI  BARBUNYA_vs_BOMBAY  \
0                 0              0                 0                   0
1                 0              0                 0                   0
2                 0              0                 0                   0
3                 1              1                 0                   0
4                 0              0                 1                   1
5                 0              0                -1                   0
6                -1              0                 0                  -1
7                 0             -1                 0                   0
8                 0              0                 1                   1
9                -1              0                 0                  -1
10                0              0                 0                   0
11                0             -1                 0                   0
12                0              0                -1                   0
13                1              1                 0                   0
14                1              1                 0                   0
15                0             -1                 0                   0
16                0              0                 0                   0

    BARBUNYA_vs_SIRA  CALI_vs_BOMBAY  CALI_vs_SIRA  BOMBAY_vs_SIRA
0                  0               0             0               0
1                  0               0             0               0
2                  0               0             0               0
3                  0               0             0               0
4                  1               0             0               0
5                  0               1             1               0
6                  0              -1             0               1
7                 -1               0            -1              -1
8                  1               0             0               0
9                  0              -1             0               1
10                 0               0             0               0
11                -1               0            -1              -1
12                 0               1             1               0
13                 0               0             0               0
14                 0               0             0               0
15                -1               0            -1              -1
16                 0               0             0               0

[17 rows x 28 columns]
```

Manual Methods

Standard method:-

Identify Unique Classes

Unique classes: SEKER, DERMASON, HOROZ, BARBUNYA, CALI, BOMBAY, SIRA

Create Binary Variables

For each unique class, create a new binary variable.

- SEKER: 1 if observation is SEKER, 0 otherwise.
- DERMASON: 1 if observation is DERMASON, 0 otherwise.
- HOROZ: 1 if observation is HOROZ, 0 otherwise.
- BARBUNYA: 1 if observation is BARBUNYA, 0 otherwise.
- CALI: 1 if observation is CALI, 0 otherwise.
- BOMBAY: 1 if observation is BOMBAY, 0 otherwise.
- SIRA: 1 if observation is SIRA, 0 otherwise.

SEKER	DERMASON	HOROZ	BARBUNYA	CALI	BOMBAY	SIRA
1	0	0	0	0	0	0
0	1	0	0	0	0	0
1	0	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	1
0	0	0	0	0	0	1
0	0	0	0	0	1	0
0	1	0	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	1	0	0

0	0	1	0	0	0	0
0	0	1	0	0	0	0

Error-correcting code method

Identify Unique Classes

- SEKER
- DERMASON
- HOROZ
- BARBUNYA
- CALI
- BOMBAY
- SIRA

Assign Binary Codes

- SEKER: 000
- DERMASON: 001
- HOROZ: 010
- BARBUNYA: 011
- CALI: 100
- BOMBAY: 101
- SIRA: 110

Encode Data

A	P	L	I	K	Ec	C	Ed	Encoded_Class
31823	663	223	182	1.23	0.58	32274	201	000
27275	605	220	158	1.39	0.69	27604	186	001
32799	654	220	190	1.16	0.5	33087	204	000
58434	981	396	190	2.09	0.88	59309	273	010
68513	1015	359	244	1.47	0.73	69406	295	011
85702	1107	428	257	1.66	0.8	86542	330	100
137358	1365	508	345	1.47	0.73	138093	418	101
41643	769	295	181	1.63	0.79	42233	230	110
68551	1025	356	246	1.45	0.72	69684	295	011
137115	1427	519	337	1.54	0.76	138970	418	101
27277	605	218	159	1.37	0.68	27611	186	001
41646	762	286	186	1.53	0.76	42074	230	110
85666	1119	436	251	1.73	0.82	86305	330	100
58454	965	392	196	2	0.87	60280	273	010

58484	956	382	197	1.94	0.86	59456	273	010
41646	768	288	186	1.55	0.76	42225	230	110
27267	597	215	162	1.33	0.66	27575	186	001

Nested dichotomies

Given classes

- SEKER
- DERMASON
- HOROZ
- BARBUNYA
- CALI
- BOMBAY
- SIRA

Create Binary variable for each class

Sno	A	P	L	I	K	Ec	C	Ed	SEKER	DERMASON	HOROZ	BARBUNYA	CALI	BOMBAY	SIRA
0	31823	663	223	182	1.2	0.6	32274	201	1	0	0	0	0	0	0
1	27275	605	220	158	1.4	0.7	27604	186	0	1	0	0	0	0	0
2	32799	654	220	190	1.2	0.5	33087	204	1	0	0	0	0	0	0
3	58434	981	396	190	2.1	0.9	59309	273	0	0	1	0	0	0	0
4	68513	1015	359	244	1.5	0.7	69406	295	0	0	0	1	0	0	0
5	85702	1107	428	257	1.7	0.8	86542	330	0	0	0	0	1	0	0
6	137358	1365	508	345	1.5	0.7	138093	418	0	0	0	0	0	1	0
7	41643	769	295	181	1.6	0.8	42233	230	0	0	0	0	0	0	1
8	68551	1025	356	246	1.5	0.7	69684	295	0	0	0	1	0	0	0
9	137115	1427	519	337	1.5	0.8	138970	418	0	0	0	0	0	1	0
10	27277	605	218	159	1.4	0.7	27611	186	0	1	0	0	0	0	0
11	41646	762	286	186	1.5	0.8	42074	230	0	0	0	0	0	0	1
12	85666	1119	436	251	1.7	0.8	86305	330	0	0	0	0	1	0	0
13	58454	965	392	196	2	0.9	60280	273	0	0	1	0	0	0	0
14	58484	956	382	197	1.9	0.9	59456	273	0	0	1	0	0	0	0
15	41646	768	288	186	1.6	0.8	42225	230	0	0	0	0	0	0	1
16	27267	597	215	162	1.3	0.7	27575	186	0	1	0	0	0	0	0

We'll partition the classes into two groups in alphabetical order.

To apply nested dichotomies, split the classes into binary classification tasks.

First Split:

- Group 1: SEKER, DERMASON, HOROZ
- Group 2: BARBUNYA, CALI, BOMBAY, SIRA

Create a binary column split_1:

- split_1 = 1 if the class is in Group 1
- split_1 = 0 if the class is in Group 2

Second Split for Group 1:

- Sub-group 1.1: SEKER
- Sub-group 1.2: DERMASON, HOROZ

Create a binary column split_1_1:

- split_1_1 = 1 if the class is SEKER
- split_1_1 = 0 if the class is DERMASON or HOROZ

Second Split for Group 2:

- Sub-group 2.1: BARBUNYA, CALI
- Sub-group 2.2: BOMBAY, SIRA

Create a binary column split_2:

- split_2 = 1 if the class is in Sub-group 2.1
- split_2 = 0 if the class is in Sub-group 2.2

A	P	L	I	K	Ec	C	Ed	SEKER	DERMASON	HOROZ	BARBUNYA	CALI	BOMBAY	SIRA	split_1	split_1_1	split_2
31823	663	223	182	1.2	0.6	32274	201	1	0	0	0	0	0	0	1	1	0
27275	605	220	158	1.4	0.7	27604	186	0	1	0	0	0	0	0	1	0	0
32799	654	220	190	1.2	0.5	33087	204	1	0	0	0	0	0	0	1	1	0
58434	981	396	190	2.1	0.9	59309	273	0	0	1	0	0	0	0	1	0	0
68513	1015	359	244	1.5	0.7	69406	295	0	0	0	1	0	0	0	0	0	1
85702	1107	428	257	1.7	0.8	86542	330	0	0	0	0	1	0	0	0	0	1
137358	1365	508	345	1.5	0.7	138093	418	0	0	0	0	0	1	0	0	0	0
41643	769	295	181	1.6	0.8	42233	230	0	0	0	0	0	0	1	0	0	0
68551	1025	356	246	1.5	0.7	69684	295	0	0	0	1	0	0	0	0	0	1
137115	1427	519	337	1.5	0.8	138970	418	0	0	0	0	0	1	0	0	0	0
27277	605	218	159	1.4	0.7	27611	186	0	1	0	0	0	0	0	1	0	0
41646	762	286	186	1.5	0.8	42074	230	0	0	0	0	0	0	1	0	0	0
85666	1119	436	251	1.7	0.8	86305	330	0	0	0	0	1	0	0	0	0	1
58454	965	392	196	2	0.9	60280	273	0	0	1	0	0	0	0	1	0	0
58484	956	382	197	1.9	0.9	59456	273	0	0	1	0	0	0	0	1	0	0
41646	768	288	186	1.6	0.8	42225	230	0	0	0	0	0	0	1	0	0	0
27267	597	215	162	1.3	0.7	27575	186	0	1	0	0	0	0	0	1	0	0