Student Name:_____Roll No:_____Section:_____

CS334 - Machine Learning

Lab 10 - Probability Based Learning

Instructor: Ms. Maham Ashraf
E-mail: mashraf@uit.edu

Semester: Fall, 2023

Objective

The purpose of this lab session is to introduce probability based learning models, such as Naive-Bayes classifier for categorical and continues features, and Naive Bayes classifier with Laplace Smoothing.

Instructions

You have to perform the following tasks yourselves. Raise your hand if you face any difficulty in understanding and solving these tasks. **Plagiarism** is an abhorrent practice and you should not engage in it.

How to Submit

- Submit lab work in a single .py file on Microsoft Teams. (No other format will be accepted)
- Lab work file name should be saved with your roll number (e.g. 19a-001-SE_LW01.py)
- Submit home work in a single .py file on Microsoft Teams. (No other format will be accepted)
- Home work file name should be saved with your roll number (e.g. 19a-001-SE_HW01.py)

1 The Naive Bayes Model - Standard Approach

A naive Bayes model returns a **MAP** prediction where the posterior probabilities for the levels of the target feature are computed under the assumption of **conditional independence** between the descriptive features in an instance given a target feature level. More formally, the **naive Bayes model** is defined as

$$\mathrm{M}(q) = \underset{l \in levels(t)}{\arg\max} \prod_{i=1}^{m} P(q[i]|t = l) \times P(t = l$$

where t is a target feature with a set of levels, $levels(t)$, and q is a query instance with a set of descriptive features, $q[1], \ldots, q[m]$.
Now we develop a naive base classifier for categorical data sets. The following steps we will follow to classify a query instance from the categorical data set.

1. Create a python class and initialize with two parameters; data set in data frame, and target variable name in data frame.
2. Compute the probabilities of each level of the target variable.

3. Compute the conditional probabilities of a descriptive feature on each level.
4. Store all the conditional probabilities of all features for classifying a query instance.
5. Write function that fit your model on Naive Bayes classifier.
6. Write a function that predict a new query instance.
7. Write a function that returns all probabilities values that your model use for prediction.
8. Implement Laplace Smoothing method for avoiding zero probability error.

$$P(f = v|t) = \frac{count(f=v|t) \times k}{count(f|t) + (k \times |Domain(f)|)}$$

2 Implementation of Naive Bayes algorithm

In this section, some hints are provided to implement Naive Bayes classifier as defined in Section - 1.
Step 1: Create a python class and initialize with two parameters; data set in data frame, and target variable name in data frame.

```
class NBClassifier:
    def __init__(self,data,target):
        self.data = data
        self.target = target
```

Step 2: Compute the probabilities of each level of the target variable.

```
def target_prob(self):

        for c in class_list: #for each class in the label
                total_class_count = self.data[self.data[self.target] == c].
    shape[0] #number of the class
                self.tprob_dict[c]=total_class_count/total_rows
        return self.tprob_dict
```

Step 3: Compute the conditional probabilities of a descriptive feature on each level.

```
def cond_prob(self,feature):
        counter=0
        for t_level in target_class_list:
            t_level_count = self.data[self.data[self.target] == t_level].
    shape[0]
            for f_level in feature_class_list:
                counter=0
                for i in range(total_rows):
                    if (self.data.loc[i,feature] == f_level) and (self.data.
    loc[i,self.target] == t_level):
                        counter += 1
                cond_prob_list.append((feature,f_level,t_level,round(counter
    / t_level_count,3)))
        return cond_prob_list
```

Step 4: Store all the conditional probabilities of all features for classifying a query instance.

```
def count_all_prob(self,k=None):
        for feature in X.columns:
                cond_list = self.cond_prob(feature)
                self.prob_all = self.prob_all + cond_list
        return self.prob_all
```

Step 5: Write function that fit your model on Naive Bayes classifier.

```
def fit():
        self.count_all_prob()
        return None
```

Step 6: Write a function that predict a new query instance.

```
def pred_score(self,query):
        if query_length != num_of_arg:
            print('Query required ',num_of_arg,' number of arguments')
            print('Data set column',self.data.columns)
        else:
```

```
            i=0
            for item in query:
                if item not in self.data.iloc[:,i].unique():
                    print(item,' is not in coulmn ',self.data.columns[i])
                i+=1
            for t_level in self.data[self.target].unique():
                for cnt in range(query_length):
                    for item in self.prob_all:
                        if item[0]==feature_list[cnt] and item[1]==query[cnt
    ] and item[2]==t_level:
                            prob_prod = prob_prod * item [ 3 ]
                prob_dict [ t_level ]= round ( prob_prod * self . tprob_dict [ t_level
    ],4)
                prob_prod =1
        return prob_dict

def pred(self,query):
        pr_dict = self . pred_score ( query )
        return max(pr_dict , key=pr_dict.get)
```

Step 7: Write a function that returns all probabilities values that use your model for prediction.

```
def get_prob_dict(self):
        print ( ' Target probabilities : ', self . tprob_dict )
        print('Conditional features Probabilities: ',self.prob_all)
        return None
```

Step 8: Implement Laplace Smoothing method for avoiding zero probability error.

```
def cond_prob_LPSmoother(self,feature,k):
        counter=0
        f_level_count = len(feature_class_list)
        for t_level in target_class_list:
            t_level_count = self.data[self.data[self.target] == t_level].
    shape[0]
            for f_level in feature_class_list:
                counter=0
                for i in range(total_rows):
                    if (self.data.loc[i,feature] == f_level) and (self.data.
    loc[i,self.target] == t_level):
                        counter+=1
                cp_smoother = (counter + k)/(t_level_count+(k*f_level_count)
    )
                cond_prob_list.append((feature,f_level,t_level,round(
    cp_smoother,3)))
        return cond_prob_list
```

Your NBClassifier class should work like as,

```
data = pd.read_csv('LoanApplication_cat.csv',sep=',')
df = pd.DataFrame(data)
nbc = NBClassifier(df,'Fraud')
nbc.fit()
query = ['paid','none','rent']
nbc.pred_score(query)
```

and also like this,

```
nbc = NBClassifier(df,'Fraud')
nbc.fit(smoothing='LPS',k=3)
query = ['paid','none','rent']
nbc.pred(query)
```

it should returns all probabilities by calling function `get_prob_dict()`,

```
nbc = NBClassifier(df,'Fraud')
nbc.fit(smoothing='LPS',k=3)
nbc.get_prob_dict()
```

the output of the above code will print all the probabilities as,

Target probabilities: True: 0.3, False: 0.7
Conditional features Probabilities: [('CrdtHistory', 'current', True, 0.333), ('CrdtHistory', 'paid', True, 0.222), ('CrdtHistory', 'arrears', True, 0.222), ('CrdtHistory', 'none', True, 0.222), ('CrdtHistory', 'current', False, 0.269), ('CrdtHistory', 'paid', False, 0.269), ('CrdtHistory', 'arrears', False, 0.346), ('CrdtHistory', 'none', False, 0.115), ('GCoApplicant', 'none', True, 0.533), ('GCoApplicant', 'guarantor', True, 0.267), ('GCoApplicant', 'coapplicant', True, 0.2), ('GCoApplicant', 'none', False, 0.652), ('GCoApplicant', 'guarantor', False, 0.13), ('GCoApplicant', 'coapplicant', False, 0.217), ('Accommodation', 'own', True, 0.467), ('Accommodation', 'rent', True, 0.333), ('Accommodation', 'free', True, 0.2), ('Accommodation', 'own', False, 0.609), ('Accommodation', 'rent', False, 0.217), ('Accommodation', 'free', False, 0.174)]

3 Lab Tasks

1. Use the hints as provided in Section -2 for implementing Naive Bayes classifier for categorical variable and implement a class NBClassifier.
2. Modify your NBClassifier that can work on continues descriptive and target variables.

4 Homework - Project Assignment - 2.1

Write a report on ML_Lab08 home work task that should answer the following questions.

1. How many irrelevant features have you discovered in Respiratory Cancer data set (e.g. RESPIR.csv). Write their names and logical reasons for considering them as an irrelevant feature.
2. How many Redundant features have you discovered in Respiratory Cancer data set. Write their names and logical reasons for considering them as a redundant feature. Did you take help from any statistical or ML method for determining the redundancy in the features?
3. How many Interacting features have you discovered in Respiratory Cancer data set. Write their names and logical reasons for considering them as an Interacting feature. Did you take help from any statistical or ML method for determining the interaction among the features?
4. How many predictive features have you discovered in Respiratory Cancer data set. Write their names and logical reasons for considering them as an predictive feature. Did you take help from any statistical or ML method for determining the predictive features features?
5. Discuss the implementation details of Bagging and Boosting models for Respiratory Cancer data set.
6. Discuss the results in terms of accuracy, ROC curve, confusion matrix, and F1 measures.