Laurence Kim

COEN 281

11/13/23

Assignment 2: Practice Supervised Learning Using Decision Trees and Naive Bayes Classifier

SECTION 1: Train and Inspect Models

1. Multinomial Naive Bayes Classifier

My procedure to train the Multinomial Naive Bayes Classifier was to first import the adult.csv file and then remove the entries with missing values. After doing so, I encoded the features that were categorical predictors as numerical input data is a friendly input for the model that we are implementing. Afterwards, I split the dataset into the categorical predictor classes as the input features and the income class as the Y output variable . Then, I fit the Multinomial NB classifier with X and y.

The categorical features that I used to train the Multinomial Naive Bayes Classifier are ['workclass', 'education', 'marital.status', 'occupation', 'relationship', 'race', 'sex', 'native.country']. My evidence to prove the predictive quality of the categorical features was to calculate the importance scores of said features. The sklearn library makes it easy to calculate the score by making a simple function call. With the importance score, I found the absolute difference in feature log probabilities between the classes and chose the 4 highest scores in magnitude which show the most influential features that result in our class predictions in the Naive Bayes classifier.

2. Gaussian Naive Bayes Classifier

The numerical features that I used to train the Gaussian Naive Bayes Classifier are ['age', 'fnlwgt', 'education.num', 'capital.gain', 'capital.loss', 'hours.per.week']. I standardized the numeric predictor values as float types and then split the X and y in a similar fashion as I did in part 1. My numerical features this time were the predictors and the income stayed as my target variable. My evidence to prove the predictive quality of the numerical features was to calculate the absolute differences between the two income groups to consider the more influential in making income predictions. I sorted the power scores in descending order and chose the top three with highest absolute differences in means to predict which three numeric features had the most influence in predicting the income values.

3. Decision Tree classifier

Some good takeaways from observing this tree is that the key features of determining income include capital gains and losses, age, hours worked per week, and education level. Furthermore, when we see that the first split on the tree is capital gain, we see that its one of the most important predictors of income. As we travers down the tree, the splits are becoming much more and more specific combining multiple features to refine the prediction. As we have learned in lecture the leaf nodes are the final predictions being that the person has a <=50K salary or >50k salary. I was not sure what the color would be representing, but upon research, I learned what they represented. We can observe that the saturation of the nodes represent the class purity, with lighter nodes signifying a mixture of classes and the darker ones contrasting as more pure distinctions. The previous classifiers that we have implemented do not provide such a visual medium, but this one is very easy to grasp the main takeaways at a glance.

The tree's visualization as seen below shows the decision paths of the classifier using Gini criteria to predict whether a person's income would be below or above $50k. The gini split can be understood as a lower score meaning a better split of the tree. I chose a max_depth of 5 which restricted the tree's levels to only 5. I know that too many levels can result in overfitting and smaller depth can result in simplification. The min samples split was set to 20 which means that the node will be split only if it contains more than 20 samples to help generalize the model with samples' tendencies that take precedence in the dataset. Finally, the min samples leaf was set to 10 to insure that we do not fall into losing accuracy from noise and making the model more stable.

SECTION 2: Compare Learned Models On Predictive Accuracy (7 pts)

I did not use any specific hyper parameters for the Naive Bayes classifiers and Decision Tree classifiers that were required of us.

There are 4 main curves to address. First, we see that the Multinomial NB has an AUC of 0.73 which is okay. It is probably most effective only when using categorical features. Second, we see that the Gaussian NB has an AUC of 0.82 which is the highest of the bunch. It is our best bet and will perform greatly when used for continuous or normally distributed data like we trained with the numerical predictors. Then, we see the combination of the MNB and the GNB which is a more holistic approach to predict the output variable that is the salary. It stands obviously better than the MNB alone, but worse than the GNB. Finally, the decision tree is also mediocre as a better approach than randomly guessing, but it stands to struggle with non-linear models that may likely contain complex patterns that are above it. In conclusion, we see the GNB to yield the best results. The diagonal dashed line in the graph is the reference to compare how useful each classifier is as we see a curve further away to have the best ability to distinguish the salary distinction apart.

SECTION 3: APPENDIX AND CODE

Top 4 Categorical Predictors for Salary

```
Predictive Feature: relationship, Importance Score: 0.8337683007134977
Predictive Feature: sex, Importance Score: 0.31717143985997964
Predictive Feature: marital.status, Importance Score: 0.2847837386533101
Predictive Feature: occupation, Importance Score: 0.07406288211375589
```

Code Part 1:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc




# Load the dataset
data = pd.read_csv("adult.csv")

# Remove rows entries with missing values
data = data.replace('?', pd.NA).dropna()
predictors= data.columns.drop('income')

categoricalPredictors = ['workclass', 'education', 'marital.status',
'occupation', 'relationship', 'race', 'sex', 'native.country']
```

```
# Encoding
label_encoders = {}
for feature in categoricalPredictors:
    le = LabelEncoder()
    data[feature] = le.fit_transform(data[feature])
    label_encoders[feature] = le

# Split X and y
X = data[categoricalPredictors]
y = data['income']

# Multinomial NB classifier
clf = MultinomialNB()
clf.fit(X, y)

# Feature importance calculation
feature_log_probs = clf.feature_log_prob_
feature_importance_scores = abs(feature_log_probs[1] -
feature_log_probs[0])
top_features_indices = feature_importance_scores.argsort()[::-1][:4]

# Top four predictive features
top_features = [list(label_encoders.keys())[i] for i in
top_features_indices]
for feature, score in zip(top_features,
feature_importance_scores[top_features_indices]):
    print(f"Predictive Feature: {feature}, Importance Score: {score}")
```

Part 2 Code:

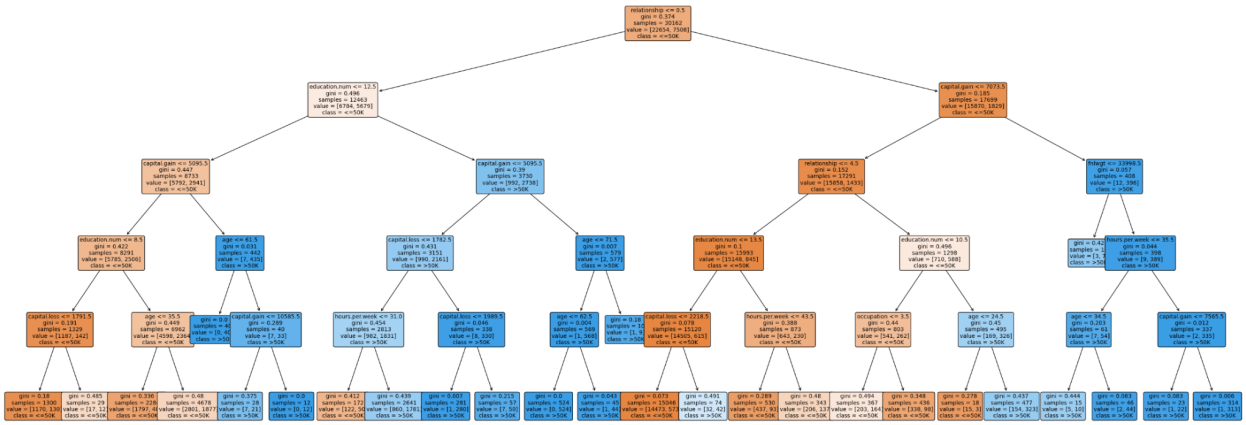
```
# Load the dataset
```

```
numericPredictors = ['age', 'fnlwgt', 'education.num', 'capital.gain',
'capital.loss', 'hours.per.week']


data[numericPredictors] = data[numericPredictors].astype(float)


# Split X and y
X = data[numericPredictors]
y = data['income']


# Gaussian NB classifier
clf = GaussianNB()
clf.fit(X, y)


means = clf.theta_


predictive_power = means[1] - means[0]  # '>50K' vs '<=50K'


top_features_indices = predictive_power.argsort()[::-1][:3]
top_features = [numericPredictors[i] for i in top_features_indices]


# Top predictive numeric features
for feature in top_features:
    print(f"Predictive Feature: {feature}")
```

Top 3 Numerical Predictors for Salary

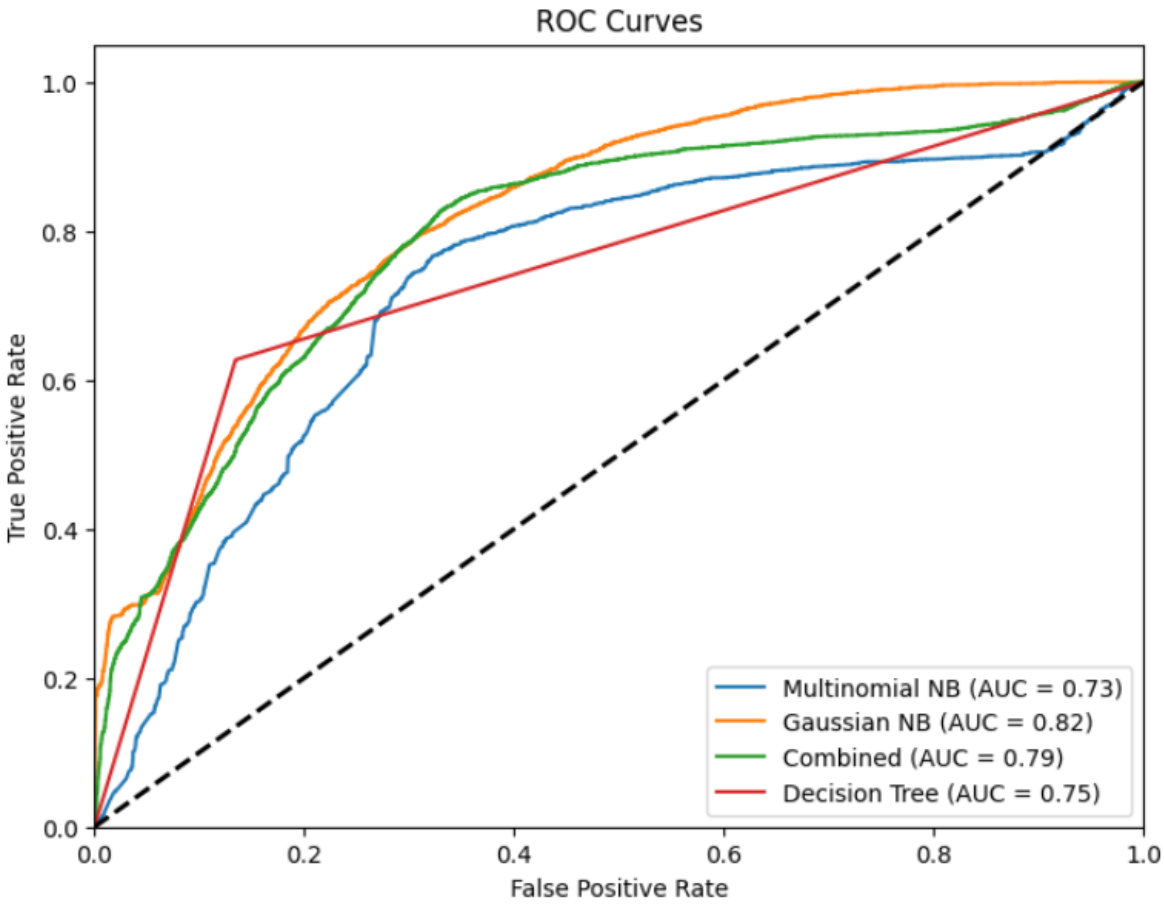
```
Predictive Feature: capital.gain
Predictive Feature: capital.loss
Predictive Feature: age
```

Binary Tree Classifier Visual output



Part 3 Code:

```
X = data.drop('income', axis=1)

y = data['income']


# Train the Decision Tree classifier

clf = DecisionTreeClassifier(max_depth=5, min_samples_split=20,
min_samples_leaf=10)

clf.fit(X, y)


# Plot tree

plt.figure(figsize=(40, 15))

plot_tree(clf, filled=True, feature_names=X.columns, class_names=['<=50K',
'>50K'], rounded=True, fontsize=10)

plt.show()
```


ROC Curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Multinomial NB (AUC = 0.73)
Gaussian NB (AUC = 0.82)
Combined (AUC = 0.79)
Decision Tree (AUC = 0.75)