Name: Shrey Pendurkar
Class: D15C
Batch: C
Roll No: 64

DMBI - Experiment 4

Aim: To implement classification algorithms Decision Tree and Naive Bayes's Algorithm using python.

Theory:

Classification is a type of **supervised machine learning** where the goal is to predict the **class label** (or category) of given input data based on training from historical data.

The typical workflow:

1. **Define target classes**: e.g., High vs Low Earnings, or Popular vs Less Popular channels.
2. **Select features**: Use metrics like uploads, views, socio-economic indicators, creation date, etc.
3. **Split data** into training and test sets.
4. **Train classification models** (Decision Tree, Naive Bayes).
5. **Evaluate** with metrics like Accuracy, Precision, Recall, F1-Score, and Confusion Matrix.

In this context, classification might help you understand:

- Do socio-economic factors (like urban population or education) predict channel success?
- Can behavioural metrics (like uploads, views, subscriber growth) categorize content performance tiers?

Applying the Algorithms

1. Decision Tree Classifier

- **How it works**: Splits data recursively on the most informative features (using metrics like Gini impurity or entropy) to create a tree of decision rules.

- **Pros**:

    - Easy to interpret—each decision node highlights which feature splits matter.
    - Can incorporate both numerical (views, uploads) and categorical data (category, channel type).

- **Cons**:

    - Prone to overfitting if not tuned (use `max_depth`, `min_samples_leaf`, etc.).
    - Can be unstable—small data changes may alter the tree.

2. Naive Bayes Classifier

- **How it works**: Applies Bayes' Theorem under the assumption that all features are independent of each other given the class label.

- **Variants**:

    - **GaussianNB** → assumes continuous features follow a normal distribution.
    - **CategoricalNB** → works for categorical features, but expects non-negative integer input.

- **Pros**:

    - Simple, fast to train—even on large datasets.
    - Effective for high-dimensional data when independence roughly holds.

- **Cons**:

    - Unrealistic independence assumption can hurt accuracy.
    - Continuous features may need transformation or binning.

Conclusion:

Implementing Decision Tree and Naive Bayes classifiers provided a clear understanding of classification workflows, from data preparation to model evaluation . This practical illustrated how Decision Trees use feature splits to create interpretable models, while Naive Bayes applies simple probabilistic reasoning, each with distinct advantages and limitations.

Key Takeaways

- Hands-On Data Challenges:
  Processing the data highlighted issues such as missing values and the need for careful feature encoding, which are crucial for accurate classification .
- Model Performance:
  Decision Trees risked overfitting without proper tuning of parameters, while Naive Bayes occasionally struggled due to its independence assumption between features .
- Metric-Based Evaluation:
  Relying on metrics like accuracy and F1-score demonstrated the importance of evaluating models beyond just their initial output, guiding improvements with each iteration .

Personal Experience:

Facing these challenges deepened understanding of both the flexibility of Decision Trees and the speed and simplicity of Naive Bayes. The practical reinforced the need for thorough data preparation and showcased the iterative nature of developing effective machine learning models .

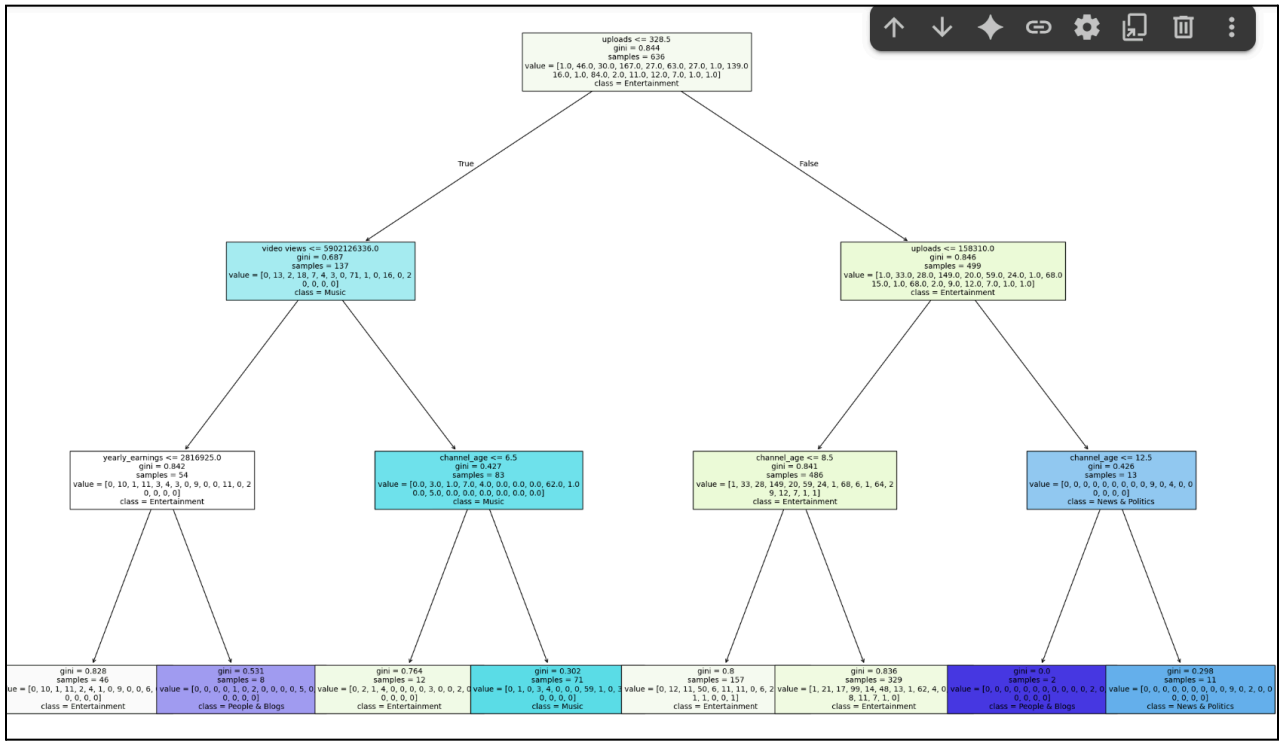
Challenges Faced:

- Data Preparation:
  Preparing the dataset for both algorithms was challenging. Handling missing values, transforming categorical data, and normalizing/encoding features required careful preprocessing before feeding them into models .
- Overfitting in Decision Trees:
  The Decision Tree model tended to overfit on the training data, especially with deep trees. Addressing this required hyperparameter tuning (such as adjusting max_depth or min_leaf) and sometimes pruning less informative branches.

Output:

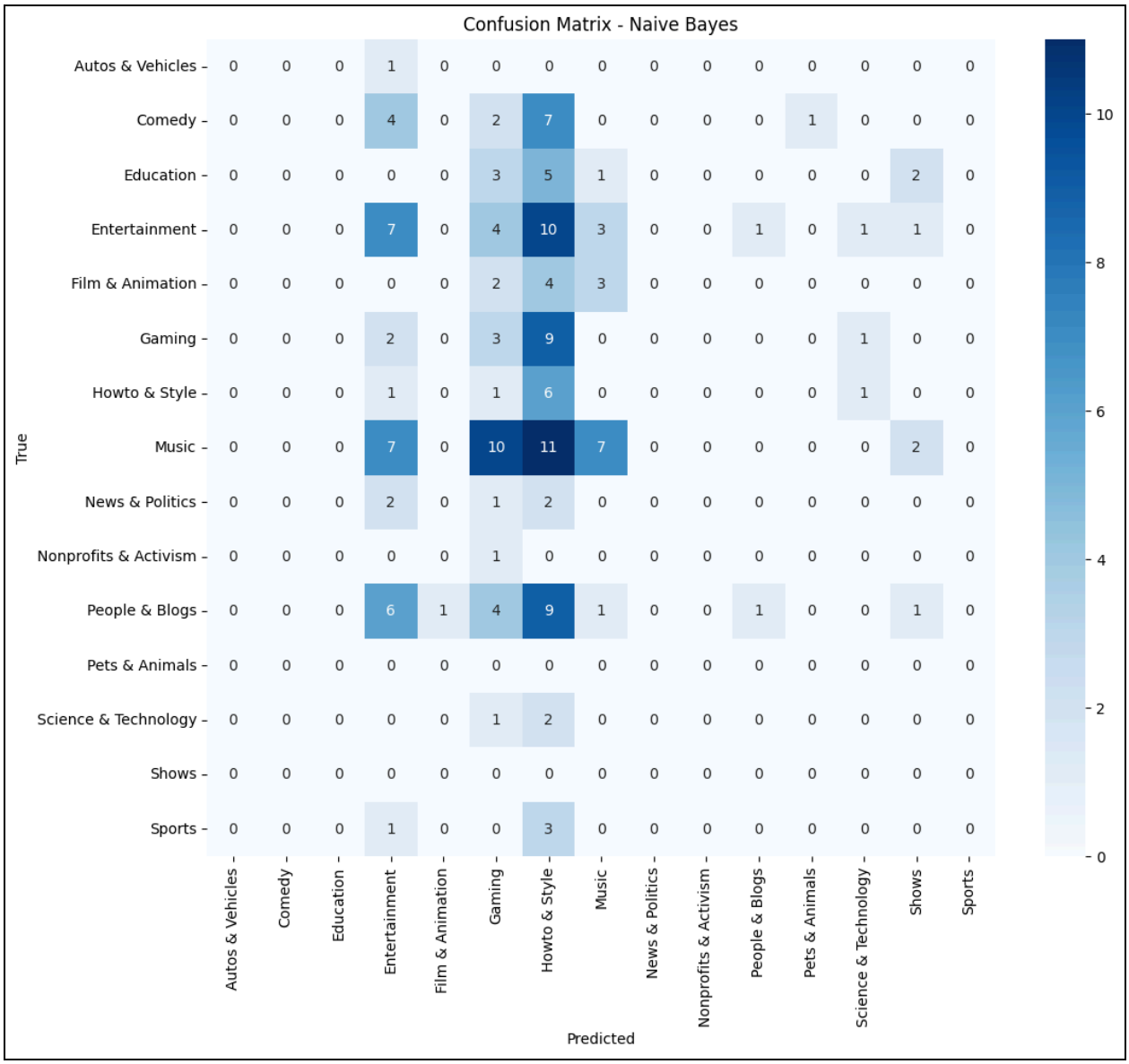
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

features = ['subscribers', 'video views', 'uploads', 'yearly_earnings', 'channel_age', 'asgpy']
target = 'category'
df_classified = df.dropna(subset=features + [target]).copy()

le = LabelEncoder()
df_classified['category_encoded'] = le.fit_transform(df_classified[target])
X = df_classified[features]
y = df_classified['category_encoded']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
dt_classifier = DecisionTreeClassifier(max_depth=3, random_state=42)
dt_classifier.fit(X_train, y_train)

y_pred = dt_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the Decision Tree classifier with max_depth=3: {accuracy:.2f}")
plt.figure(figsize=(30,20))
plot_tree(dt_classifier, filled=True, feature_names=features, class_names=le.classes_.tolist(), fontsize=10)
plt.show()
```

```
Accuracy of the Decision Tree classifier: 0.87
```



```
df.drop(df[df['highest_yearly_earnings'] < 100].index, inplace=True)
df.drop(df[df['lowest_yearly_earnings'] < 100].index, inplace=True)

df['yearly_earnings'] = (df['lowest_yearly_earnings'] + df['highest_yearly_earnings']) / 2
df = df[df['created_year'] != 1970]
df.dropna(subset=['category'], inplace=True)
df = df[df['video views'] != 0]
df['channel_age'] = 2023 - df['created_year']
df['asgpy'] = df['subscribers']/df['channel_age']


features = ['subscribers', 'video views', 'uploads', 'yearly_earnings', 'channel_age', 'asgpy']
target = 'category'
df_classified = df.dropna(subset=features + [target]).copy()
le = LabelEncoder()
df_classified['category_encoded'] = le.fit_transform(df_classified[target])

X = df_classified[features]
y = df_classified['category_encoded']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)

y_pred_nb = nb_classifier.predict(X_test)
print("Classification Report - Naive Bayes (All Categories):")
unique_classes_in_test = np.unique(np.concatenate((y_test, y_pred_nb)))
target_names_subset = [le.classes_[i] for i in unique_classes_in_test]

print(classification_report(y_test, y_pred_nb, labels=unique_classes_in_test, target_names=target_names_subset, zero_division=0))
cm_nb = confusion_matrix(y_test, y_pred_nb, labels=unique_classes_in_test)

plt.figure(figsize=(12, 10))
sns.heatmap(cm_nb, annot=True, fmt='d', cmap='Blues', xticklabels=target_names_subset, yticklabels=target_names_subset)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - Naive Bayes (All Categories)')
plt.show()
```

```
Classification Report - Naive Bayes (All Categories):
                      precision    recall  f1-score   support

     Autos & Vehicles       0.00      0.00      0.00         1
               Comedy       0.00      0.00      0.00        14
            Education       0.00      0.00      0.00        11
        Entertainment       0.23      0.26      0.24        27
     Film & Animation       0.00      0.00      0.00         9
               Gaming       0.09      0.20      0.13        15
        Howto & Style       0.09      0.67      0.16         9
                Music       0.47      0.19      0.27        37
      News & Politics       0.00      0.00      0.00         5
Nonprofits & Activism       0.00      0.00      0.00         1
       People & Blogs       0.50      0.04      0.08        23
       Pets & Animals       0.00      0.00      0.00         0
 Science & Technology       0.00      0.00      0.00         3
                Shows       0.00      0.00      0.00         0
               Sports       0.00      0.00      0.00         4

             accuracy                           0.15       159
            macro avg       0.09      0.09      0.06       159
         weighted avg       0.23      0.15      0.14       159
```


Confusion Matrix - Naive Bayes
True
Predicted
Autos & Vehicles
Comedy
Education
Entertainment
Film & Animation
Gaming
Howto & Style
Music
News & Politics
Nonprofits & Activism
People & Blogs
Pets & Animals
Science & Technology
Shows
Sports
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 4 0 2 7 0 0 0 0 1 0 0 0
0 0 0 0 0 3 5 1 0 0 0 0 0 2 0
0 0 0 7 0 4 10 3 0 0 1 0 1 1 0
0 0 0 0 0 2 4 3 0 0 0 0 0 0 0
0 0 0 2 0 3 9 0 0 0 0 0 1 0 0
0 0 0 1 0 1 6 0 0 0 0 0 1 0 0
0 0 0 7 0 10 11 7 0 0 0 0 0 2 0
0 0 0 2 0 1 2 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 6 1 4 9 1 0 0 1 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 2 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 3 0 0 0 0 0 0 0 0
10
8
6
4
2
0