PATTERN RECOGNITION AND MACHINE LEARNING PROJECT

Member 1: Roll No. CS22B1075
Member 2: Roll No. CS22B1076
Member 3: Roll No. CS22B1074

Contributions:
CS22B1075: Model Building
CS22B1076: Feature Extraction
CS22B1074: Data Cleaning,PCA,Report.

Abstract

This project involves building a pipeline to process, clean, and analyze data from a dataset of inmate images. The primary goal is to estimate gender using logistic regression and predict BMI using linear regression. Various steps such as feature extraction, dimensionality reduction (PCA), and visualization are performed to derive meaningful insights.

Aim

The aim of this project is to:

- Format and clean the dataset.
- Extract features from inmate images.
- Predict gender using logistic regression and BMI using linear regression.
- Analyze and visualize results, including weight classes and charge distributions.

Approach

The approach of our project involves a structured pipeline to process and analyze inmate data, both tabular and visual, to predict relevant information such as gender and BMI. The steps in our approach are as follows:

1. **Data Formatting and Cleaning:**
   The raw CSV dataset is formatted and cleaned by fixing delimiters, removing unnecessary columns, and handling missing values to ensure high-quality input data.
2. **Feature Extraction from Images:**
   Images of inmates (front and side profiles) are processed using a pre-trained deep learning model, MobileNetV2, to extract meaningful numerical features representing image characteristics.
3. **Dimensionality Reduction:**
   Principal Component Analysis (PCA) is applied to reduce the dimensionality of the combined image features while preserving the important information.
4. **Model Training and Prediction:**
   - Logistic Regression is used to predict gender based on the extracted features.
   - Linear Regression is implemented to predict BMI values from the features.
5. **Analysis and Visualization:**
   The predicted BMI values are classified into weight categories (underweight, normal, overweight, obese), and distributions are plotted to analyze the relationship between weight classes and inmate charges.

6. **Testing with New Data:**
   The trained models are tested on new input images, such as a friend's face, to validate the performance of the pipeline.

This systematic approach integrates deep learning for feature extraction, machine learning for predictions, and data analysis for meaningful insights, forming a complete pipeline for the project.

Project Implementation

1. Formatting the CSV

The dataset provided in CSV format needed corrections due to incorrect delimiters. Below is the implementation:

```python
try:
    # Read the file with semicolon delimiter
    fixed_data = pd.read_csv(file_path, delimiter=';')
    fixed_info = fixed_data.info()  # Check the structure after fixing
    fixed_head = fixed_data.head()  # Preview the fixed content
except Exception as e:
    fixed_info = str(e)
    fixed_head = None

fixed_data.to_csv("formatted.csv")
```

Listing 1: Formatting the CSV

2. Cleaning the Dataset

Here, unnecessary columns with null values were removed, and critical missing values were handled.

```python
# Drop unnecessary columns with mostly null values
columns_to_drop = ['discharge_date', 'Unnamed: 21', 'electronic_detention_date', 'alias']
cleaned_data = fixed_data.drop(columns=columns_to_drop)

# Handle null values in important columns - dropping rows with critical missing data
essential_columns = ['id', 'name', 'date_of_birth', 'weight', 'hair', 'sex', 'height', 'race', 'eyes']
cleaned_data = cleaned_data.dropna(subset=essential_columns)

# Save the cleaned data for review

cleaned_data.to_csv("cleaned and formatted.csv",index=False)

cleaned_data.info()
```

Listing 2: Cleaning the Dataset

3. Extracting Features from Images (Front and Side)

The MobileNetV2 pre-trained model was used to extract features from images.

```

for idx, (_, row) in enumerate(id_bmi_sampled.iterrows(), start
    =1):
    img_id = row['id']
    gender = row['gender']
    bmi = row['bmi']  # Extract BMI

    # Front and side image paths
    front_path = os.path.join(images_path_front, img_id)
    side_path = os.path.join(images_path_side, img_id)

    # Preprocess images
    front_img = preprocess_image(front_path)
    side_img = preprocess_image(side_path)

    # Skip if any image is missing or cannot be processed
    if front_img is None or side_img is None:
        continue

    # Extract features using the pre-trained model
    # front_feature = feature_extractor.predict(np.expand_dims(
       front_img, axis=0))[0]
    # side_feature = feature_extractor.predict(np.expand_dims(
       side_img, axis=0))[0]

    # Extract features using the pre-trained model
    front_feature = feature_extractor.predict(np.expand_dims(
       front_img, axis=0), verbose=0)[0]
    side_feature = feature_extractor.predict(np.expand_dims(
       side_img, axis=0), verbose=0)[0]


    # Combine front and side features
    combined_feature = np.concatenate([front_feature,
       side_feature])

    # Append features, BMI labels, and ids
    features.append(combined_feature)
    bmi_labels.append(bmi)
    gender_labels.append(gender)
    ids.append(img_id)

    # Save intermediate results at intervals
    if idx % save_interval == 0:
        save_path = os.path.join(features_save_dir, f"
           features_batch_{idx // save_interval}.npz")
        np.savez(save_path, features=np.array(features),
           bmi_labels=np.array(bmi_labels), ids=np.array(ids),
           gender_labels=np.array(gender_labels))
        print(f"\nSaved batch {idx // save_interval} to {
           save_path}")
        features, bmi_labels, ids, gender_labels = [], [], [], []
             # Reset for the next batch
```

Listing 3: Extracting Features from Images

4. Applying PCA on the Dataset

Principal Component Analysis (PCA) was applied for dimensionality reduction.

```
# Apply PCA
n_components = 500  # Number of principal components to retain
pca = PCA(n_components=n_components)

# Fit PCA to the features
pca_features = pca.fit_transform(features)
```

Listing 4: Applying PCA

5. Estimating Gender using Logistic Regression

The gender was predicted using Logistic Regression.

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features,
    labels_binary, test_size=0.2, random_state=88)

# Train a Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Separate test samples by gender
X_test_male = X_test[y_test == 1]
X_test_female = X_test[y_test == 0]

# Take an equal number of male and female samples
n_samples = min(len(X_test_male), len(X_test_female))
X_test_balanced = np.vstack((X_test_male[:n_samples],
    X_test_female[:n_samples]))
y_test_balanced = np.hstack((np.ones(n_samples), np.zeros(
    n_samples)))

# Make predictions on the balanced test set
y_pred_balanced = model.predict(X_test_balanced)
```

Listing 5: Gender Prediction

5.1 Accuracy of the logistic regression model

```
Balanced Gender Prediction Accuracy: 0.97
Model saved to C:\Users\sande\Downloads\illinois_doc_dataset\front2\gender_prediction_model.pkl
Predicted Gender for Test Sample: Male
```

Listing 6: accuracy

6. BMI Prediction using Linear Regression

Linear Regression was used to predict BMI.

```python
# Load PCA-transformed features
data = np.load(pca_features_save_path)
features = data['features']
labels = data['bmi_labels']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=88)

# Train a Linear Regression model to predict BMI
model = LinearRegression()
# model = GradientBoostingRegressor(random_state=88)
# model = RandomForestRegressor(n_estimators=100, max_depth=10, random_state=42, n_jobs=-1)

model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
```

Listing 7: BMI Prediction

7. Printing All Metrics

Evaluation metrics for the models were calculated and displayed.

```
BMI Prediction Metrics:
Mean Squared Error: 17.84
Mean Absolute Error: 3.15
 R   Score: 0.35
Pearson corr: 0.59
BMI Prediction Model saved to C:\Users\sande\Downloads\illinois_doc_dataset\front2\gender_prediction_model.pkl
Predicted BMI for Test Sample: 26.32
```

Listing 8: Metrics

8. Grading BMI into Weight Classes

The predicted BMI was categorized into weight classes.

```python
def grade_bmi(bmi):
    if bmi < 18.5:
        return "Underweight"
    elif bmi < 25:
        return "Normal"
    elif bmi < 30:
        return "Overweight"
    else:
        return "Obese"

bmi_classes = [grade_bmi(bmi) for bmi in bmi_predictions]
```

Listing 9: Weight Classes

9. Plotting Distribution Between Inmates and Charges

The distribution of weight classes and inmate charges was visualized.

```python
import matplotlib.pyplot as plt

plt.hist(bmi_classes, bins=4)
plt.title("BMI Distribution")
plt.xlabel("Weight Classes")
plt.ylabel("Frequency")
plt.show()
```

Listing 10: Plotting Distribution

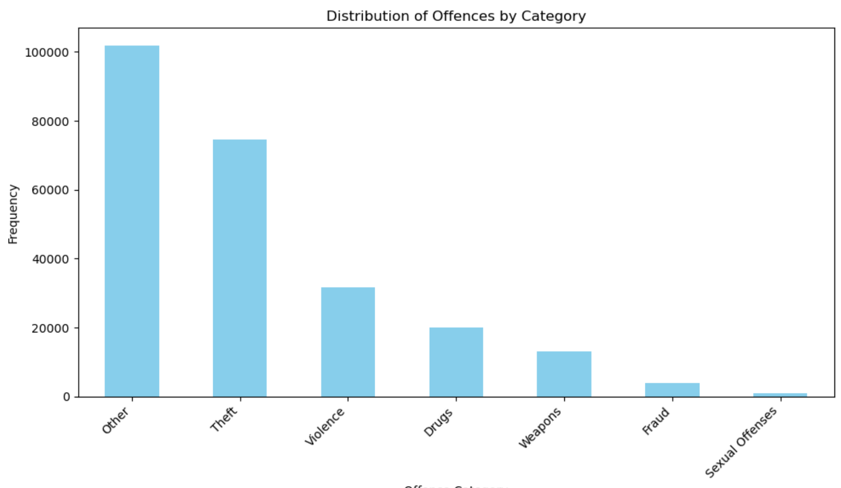

Figure 1: Distribution of offences by category.

10. Testing the Model with a Friend's Face

The trained pipeline was tested on a new image.

```python
# Preprocess front and side images
front_img = preprocess_image(images_path_front)
side_img = preprocess_image(images_path_side)


if front_img is None or side_img is None:
    print("Error: One or both images are missing or could not be
        processed.")
else:
    # Extract features using the pre-trained model
    front_feature = feature_extractor.predict(np.expand_dims(
        front_img, axis=0), verbose=0)[0]
    side_feature = feature_extractor.predict(np.expand_dims(
        side_img, axis=0), verbose=0)[0]

    # Combine front and side features
    combined_feature = np.concatenate([front_feature,
        side_feature])


me=pca.transform(np.array([combined_feature]))

grade_bmi(model.predict(me))
```

11. Predicted BMI vs Actual BMI

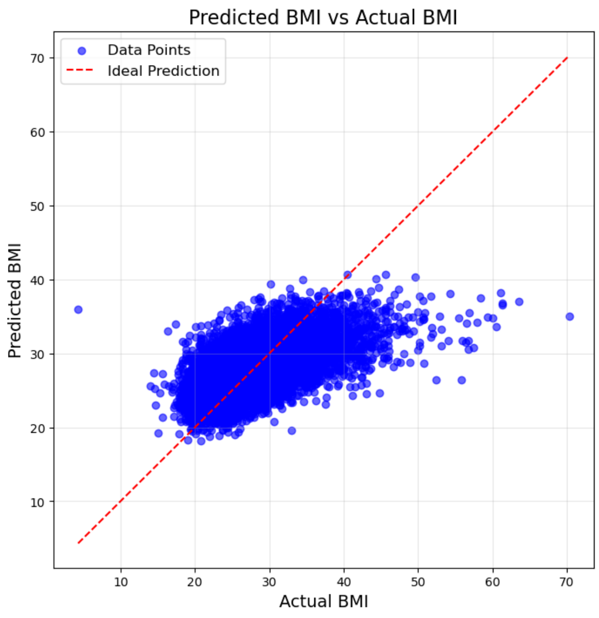

Figure 2: Distribution of offences by category.

Modules Used

- **pandas**: Data manipulation and cleaning.
- **tensorflow.keras**: Feature extraction from images.
- **sklearn**: Machine learning models and metrics.
- **numpy**: Numerical operations.
- **matplotlib**: Visualization.

References

- Scikit-Learn User Guide: `https://scikit-learn.org/`