Assignment: TABLE CLASSIFICATION FROM FINANCIAL STATEMENTS

Introduction

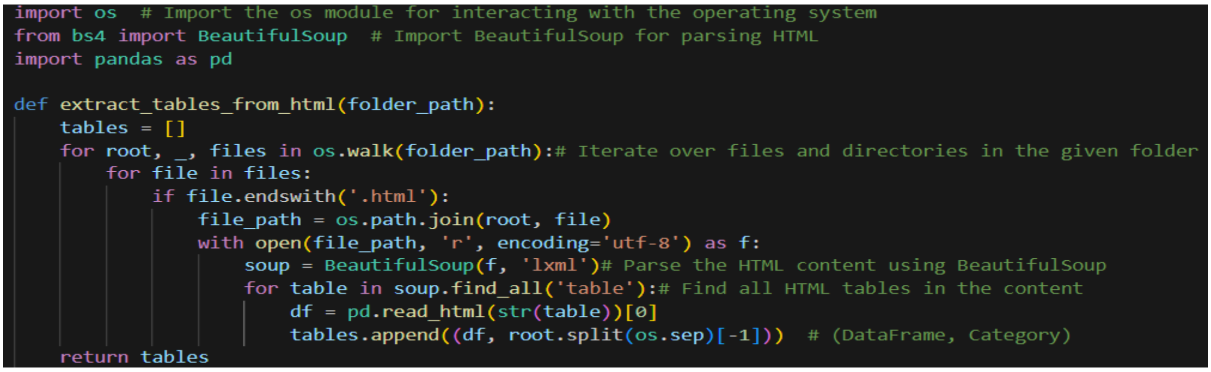
This report summarizes the approach, model selection, and results for the classification of tables from financial statements into five categories: Income Statements, Balance Sheets, Cash Flows, Notes, and Others. The dataset consists of HTML files containing tabular data, which were preprocessed and classified using machine learning techniques.

Approach

1. Data Extraction and Preprocessing

1.1. Extraction: The dataset comprises HTML files stored in five subfolders, each representing a different category. The tables were extracted from these HTML files using BeautifulSoup, OS library and pandas.

Code Snippet:

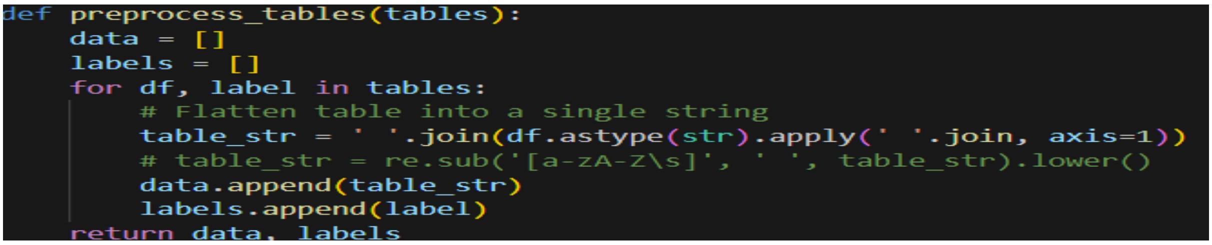


```
import os  # Import the os module for interacting with the operating system
from bs4 import BeautifulSoup  # Import BeautifulSoup for parsing HTML
import pandas as pd

def extract_tables_from_html(folder_path):
    tables = []
    for root, _, files in os.walk(folder_path):# Iterate over files and directories in the given folder
        for file in files:
            if file.endswith('.html'):
                file_path = os.path.join(root, file)
                with open(file_path, 'r', encoding='utf-8') as f:
                    soup = BeautifulSoup(f, 'lxml')# Parse the HTML content using BeautifulSoup
                    for table in soup.find_all('table'):# Find all HTML tables in the content
                        df = pd.read_html(str(table))[0]
                        tables.append((df, root.split(os.sep)[-1]))  # (DataFrame, Category)
    return tables
```

1.2. Preprocessing: The extracted tables were cleaned by removing non-alphabetic characters, converting text to lowercase, and removing extra spaces and 'nan' values.

Code Snippet:



```
def preprocess_tables(tables):
    data = []
    labels = []
    for df, label in tables:
        # Flatten table into a single string
        table_str = ' '.join(df.astype(str).apply(' '.join, axis=1))
        # table_str = re.sub('[a-zA-Z\s]', ' ', table_str).lower()
        data.append(table_str)
        labels.append(label)
    return data, labels
```

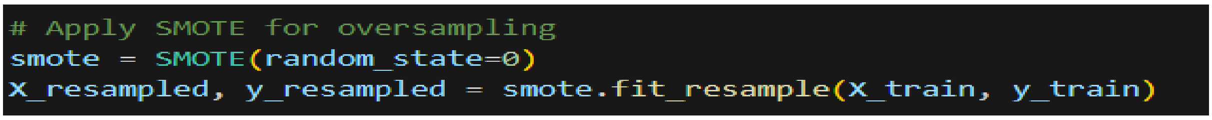
2. Feature Extraction

2.1. TF-IDF Vectorization: The text data was vectorized using TF-IDF to convert the textual information into numerical format suitable for model training.

3. Model Selection and Training

3.1. Handling Class Imbalance: SMOTE (Synthetic Minority Over-sampling Technique) was applied to address class imbalance in the dataset.

Code Snippet:



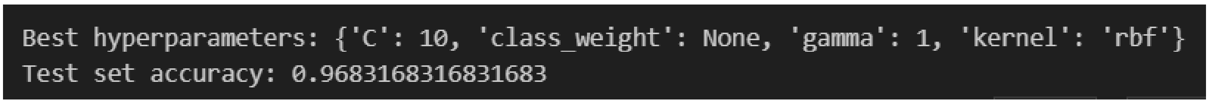
```
# Apply SMOTE for oversampling
smote = SMOTE(random_state=0)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
```

3.2. Model Training: An SVM classifier was selected for its effectiveness in text classification tasks. Hyperparameter tuning was performed using GridSearchCV to find the best parameters.

4. Model Evaluation

4.1. Accuracy Calculation: The trained model was evaluated on the test set, and accuracy was calculated to assess performance.

Code Snippet:



```
Best hyperparameters: {'C': 10, 'class_weight': None, 'gamma': 1, 'kernel': 'rbf'}
Test set accuracy: 0.9683168316831683
```

Results:

After training and hyperparameter tuning, the SVM classifier achieved an accuracy of `96.83%` on the test set. This demonstrates the model's capability to classify tables from financial statements into the specified categories effectively.

Conclusion:

The task involved extracting and preprocessing data from HTML files, feature extraction using TF-IDF, handling class imbalance with SMOTE, and training an SVM classifier with hyperparameter tuning. The final model demonstrated satisfactory performance with a high accuracy rate.