Financial Statement Classification Project Report

JUNE 2024

PREPARED BY

SUBBULAKSHMI S

ASSIGNED BY

FINCLAV

Financial StatementClassification Model

Problem Statement

This project aims to classify tables extracted from financial statements into categories: Income Statements, Balance Sheets, Cash Flows, Notes, and Others. The classification was achieved using various machine learning models, with a focus on model performance and accuracy.

About Data

The dataset consists of text documents in a form of HTML files provided in different folders representing the categories:

Balance Sheets: Contains tables of text related to balance sheets.

Cash Flow: Contains tables of text related to cash flow statements.

Income Statement: Contains tables of text related to income statements.

Notes: Contains miscellaneous tables of text related to financials.

Others: Contains tables of text that do not fit into the above categories.

Requirements

The necessary modules and libraries are imported for data reading, preprocessing, visulaizing, and for model evaluation respectively.

- pandas
- numpy
- scikit-learn
- imbalanced-learn
- matplotlib
- nltk
- beautifulsoup4

Data Extraction

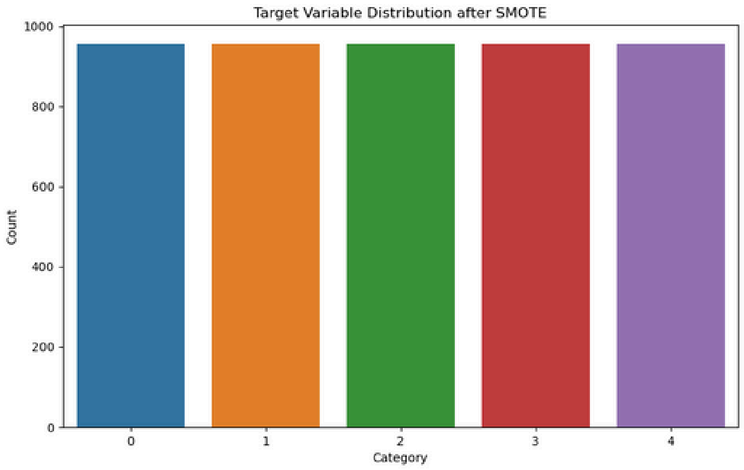
The documents were organized into folders, each representing a specific class. The extraction process involved:

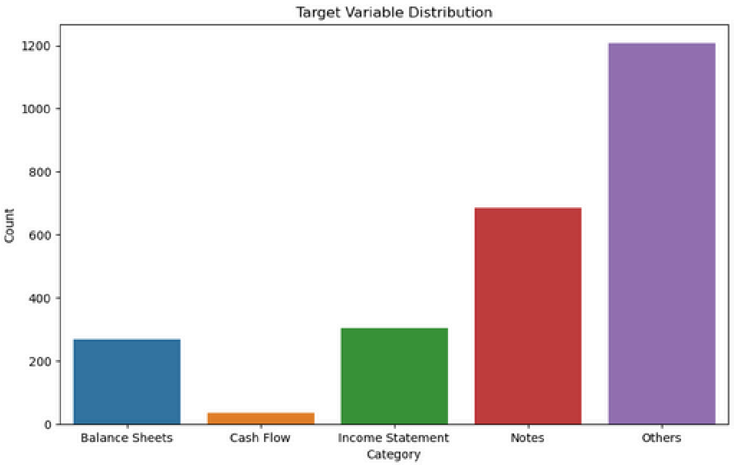
- **Reading Files**: Reading text files from each folder.
- **Labeling Data**: Assigning labels based on the folder name.
- **Combining Data**: Aggregating all documents into a single DataFrame with corresponding labels.

Data Preprocessing

- ***Text Cleaning-*** .Clean the text by removing non-alphabetic characters and extra spaces. Convert text to lowercase.
- ***Stemming-*** Apply stemming using the PorterStemmer.
- ***Label Encoder***- Encode the target variable (categories)

Class imbalance was addressed using Synthetic Minority Over-sampling Technique (SMOTE), which balances the dataset by generating synthetic samples for the minority classes.





Financial StatementClassification Model

Data Splitting and Vectorization

- Split the data into training and testing sets.
- Convert text data to numerical vectors using CountVectorizer.

Models Used

Support Vector Machine (SVM)

SVM is effective in high-dimensional spaces and is often used for text classification.

Multinomial Naive Bayes

Naive Bayes classifiers are particularly suited for text classification tasks due to their simplicity and efficiency.

Random Forest

An ensemble method that builds multiple decision trees and merges them to get a more accurate and stable prediction.

KNN

KNN is a simple instance-based learning algorithm that classifies data points based on their proximity to neighboring points,

Decision Tree

Decision Trees recursively partition the feature space to make predictions

Logistic Regression

A statistical model that in its basic form uses a logistic function to model a binary dependent variable.

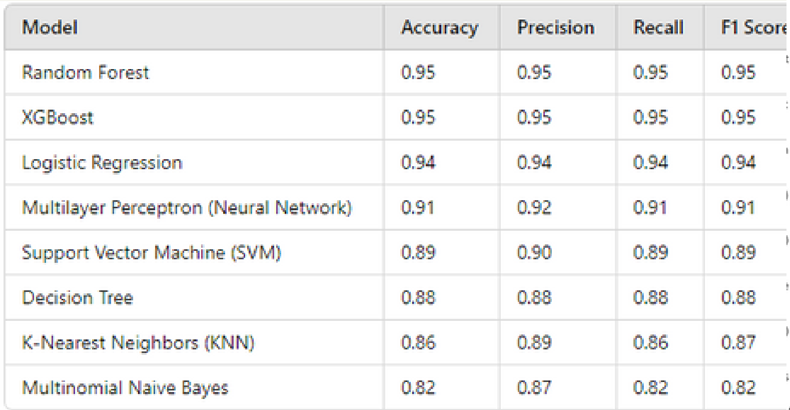
XGBoost

An optimized gradient-boosting algorithm that is highly efficient and accurate.

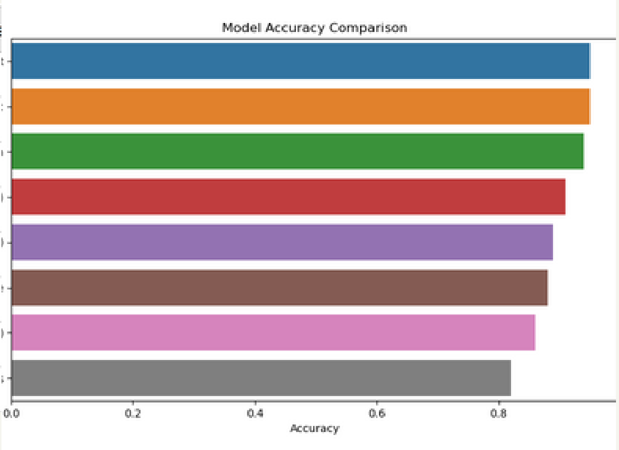
Multi Layer Perceptron

The Multilayer Perceptron (MLP) is a type of neural network that consists of an input layer, one or more hidden layers, and an output layer. The MLP uses backpropagation for training and can model complex, non-linear relationships.

Evaluation and model comparison



Model	Accuracy	Precision	Recall	F1 Score
Random Forest	0.95	0.95	0.95	0.95
XGBoost	0.95	0.95	0.95	0.95
Logistic Regression	0.94	0.94	0.94	0.94
Multilayer Perceptron (Neural Network)	0.91	0.92	0.91	0.91
Support Vector Machine (SVM)	0.89	0.90	0.89	0.89
Decision Tree	0.88	0.88	0.88	0.88
K-Nearest Neighbors (KNN)	0.86	0.89	0.86	0.87
Multinomial Naive Bayes	0.82	0.87	0.82	0.82



Conclusion

The Random Forest and XGBoost models performed the best with an accuracy of 0.95, followed by Logistic Regression with 0.94.

Given the slight class imbalance, these models showed robust performance across precision, recall, and F1 scores.

Recommendation

Given the high accuracy, precision, recall, and F1 scores of Random Forest and XGBoost, either of these models would be suitable for deployment.

Further optimization can be considered by fine-tuning hyperparameters or using more advanced techniques to handle class imbalance if needed.