DocAssist Project Report

Problem Statement:

The goal of this project is to develop an intelligent document analysis and decision support system, which helps users (such as data analysts or businesses) analyze documents efficiently. The system leverages machine learning and data analysis techniques to provide insights from documents based on their content. It processes data, identifies trends, and offers predictive recommendations that assist users in making informed decisions.

Overview:

This project builds a web application using **Flask** to facilitate document analysis. The system performs exploratory data analysis (EDA) on the provided documents, cleans and processes the data, and builds a machine learning model to make predictions based on the document contents. Random Forest is employed for classification tasks, and hyper-parameter tuning is performed for model optimization. The application is designed to provide real-time predictions and present the results through interactive visualizations, ensuring a user-friendly experience.

Libraries Used:

- **Flask**: For building the web application and handling server-side tasks.
- **pandas**: For data manipulation and analysis.
- **numpy**: For numerical operations and handling large datasets.
- **scikit-learn**: For machine learning tasks, such as data preprocessing, model training, and evaluation.
- **matplotlib**, **seaborn**, and **plotly**: For data visualization and generating interactive charts.
- **joblib**: For saving the trained machine learning models for future use.

Dataset:

The dataset consists of documents, each containing a variety of features such as text content, metadata (e.g., document type, date), and associated labels (e.g., document category). The dataset used in this project is a collection of document data, which has been preprocessed to remove unnecessary characters and perform basic text cleaning.

For the purpose of analysis and prediction, we have:

- **Text Features**: Content of the document, which will be used to classify or predict document types.
- **Metadata Features**: Includes additional information like document type, author, and timestamp.
- The dataset was cleaned and any missing values were handled before model training.

Exploratory Data Analysis (EDA):

EDA includes:

- **Visualizing Document Features**: Displaying distributions of text lengths, word frequency, and key phrases using histograms and word clouds.
- **Checking Data Integrity**: Ensuring there are no missing or duplicate entries in the dataset.
- **Visualizing Relationships**: Generating pair plots and correlation matrices to understand relationships between features.

Data Preprocessing:

- **Text Preprocessing**: Text data was tokenized, converted to lowercase, and stop words were removed to prepare the text for analysis.
- **Feature Extraction**: The text content was vectorized using TF-IDF (Term Frequency-Inverse Document Frequency) to convert the documents into numerical representations that can be used in machine learning models.
- **Categorical Features**: Any categorical features were encoded using label encoding or one-hot encoding to make them usable in machine learning models.

Model Training:

A **Random Forest Classifier** was trained on the pre-processed data. The model was evaluated using accuracy, classification report, and confusion matrix. Additionally, feature importance was analyzed to understand which words or document attributes had the most influence on predictions.

Hyperparameter Tuning:

We employed **GridSearchCV** to tune the hyperparameters of the Random Forest model. This process involved trying different combinations of parameters such as the number of trees, max depth, and min samples split to optimize model performance.

Results:

- The **Random Forest model** achieved an accuracy of approximately **85%**, with **precision**, **recall**, and **F1 score** values indicating strong predictive power.
- Feature importance analysis revealed that words with higher frequency and certain document metadata (like document type) were significant for classification.
- We also implemented a **Logistic Regression model** as a baseline, which achieved an accuracy of around **80%**.

Model Persistence:

The trained **Random Forest model** was saved using **joblib** as `docassist_model.joblib` for future use in the web application. This allows users to upload documents and get predictions without retraining the model.

Discussion of Future Work:

1. Improving Prediction Accuracy:

One avenue for improving the system would be to expand the dataset with more varied document types and higher-quality labeled data. By increasing the dataset size and diversity, we could improve the model's generalization and predictive performance.

2. Incorporating NLP Models:

We can integrate more advanced **Natural Language Processing (NLP)** models such as **BERT** or **GPT-based** models, which can provide better document understanding and improve prediction accuracy, especially in the case of complex documents.

3. User Personalization:

Currently, the system provides general predictions for documents. In the future, we could personalize the document analysis process by allowing users to input preferences or upload domain-specific documents. This would help the model to adapt and give more relevant insights based on the user's specific context.

4. Advanced Visualizations and Insights:

Adding more interactive features to the visualization, such as **interactive word clouds**, **dynamic charts**, and **real-time data filtering**, would make the analysis process more engaging for the users.

5. Integration with Other Tools:

To extend the system's utility, integrating DocAssist with popular platforms such as **Google Drive**, **Dropbox**, or **SharePoint** would make it easier to import documents directly from these services for analysis.