Project Report

Project Name: **Bank Transaction Classification Using Machine Learning**

Created By: Pooja Chaudhari

Introduction

Project Overview

The primary goal of this project is to classify bank transactions into predefined ledger categories using machine learning. This classification aims to streamline financial analysis and assist users or businesses in understanding and organizing their expenditures.

Data Preprocessing

Preprocessing was a critical step to ensure the dataset was clean, structured, and ready for machine learning model. Below are the main steps involved:

1. **Data Cleaning**:
    - Removed rows with missing or null values in essential columns such as TRANSACTION DETAILS.
    - Filled up the null values of numerical columns such as Withdrawal Amt and Deposit Amt with 0.
    - Dropped unnecessary columns such as Account No, CHQ.No. and Value Date which were not relevant for classification.
2. **Manual Annotation**
    - Manually Annotated all the transactions with different tags such as Fund Transfer, Financial Services, Salary and others.
3. **Handling Text Data**:
    - Standardized and lowercased text in the TRANSACTION DETAILS column for consistency.
    - Removed special characters and stop words to clean up noise in the text data.
4. **Feature Extraction**:
    - **TF-IDF Vectorization**:
        - Transformed the TRANSACTION DETAILS column into numerical vectors using the TF-IDF (Term Frequency-Inverse Document Frequency) method.
        - Parameters:
            - max_features=500: Limited to 500 most relevant terms.
            - ngram_range=(1, 2): Captured both single words and bigrams.
5. **Label Encoding**:
    - Encoded the Category column into numerical values using scikit-learn's LabelEncoder for compatibility with machine learning algorithms.

6. **Train-Test Split**:
    - Split the dataset into training (80%) and testing (20%) sets to evaluate model performance on unseen but labeled data.

Methodology

Model Training and Selection:

- Multiple machine learning models were trained to compare performance, including:
    - **Random Forest**: A robust ensemble model known for its ability to handle high-dimensional data.
    - **Logistic Regression**: A baseline linear model to provide insights into separability.
    - **K-Nearest Neighbors (KNN)**: A distance-based classifier for simple and interpretable predictions.
    - **Support Vector Machine (SVM)**: A powerful classifier effective in high-dimensional spaces.

Models Considered

1. **Random Forest**:
    - **Rationale**:
        - Random Forest is an ensemble model that combines multiple decision trees to improve prediction accuracy and reduce overfitting.
        - It handles high-dimensional datasets effectively and can model non-linear relationships between features and targets.
    - **Key Configurations**:
        - n_estimators=100: The number of trees in the forest.
        - random_state=42: Ensures reproducibility.
        - Default parameters for max_depth and min_samples_split.
2. **Logistic Regression**:
    - **Rationale**:
        - Logistic Regression serves as a strong baseline model, especially for linearly separable data.
        - It is computationally efficient and provides interpretable coefficients to understand feature importance.
    - **Key Configurations**:
        - max_iter=1000: Increased iteration limit to ensure convergence.
        - random_state=42: Ensures reproducibility.
        - Solver set to the default (lbfgs) for handling multiclass classification.
3. **K-Nearest Neighbors (KNN)**:
    - **Rationale**:
        - KNN is a simple and interpretable algorithm that classifies data points based on proximity to their neighbors.

        - While sensitive to feature scaling, it can perform well when decision boundaries are irregular.
    - **Key Configurations**:
        - Default settings used (k=5 neighbors).
        - No additional hyperparameters were tuned due to KNN's simplicity.
4. **Support Vector Machine (SVM)**:
    - **Rationale**:
        - SVM is a powerful classifier that finds the optimal hyperplane to separate classes in a high-dimensional space.
        - Effective when classes are not linearly separable due to its kernel trick.
    - **Key Configurations**:
        - Kernel: Linear (to reduce complexity and speed up training).
        - random_state=42: Ensures reproducibility.

Model Training Process

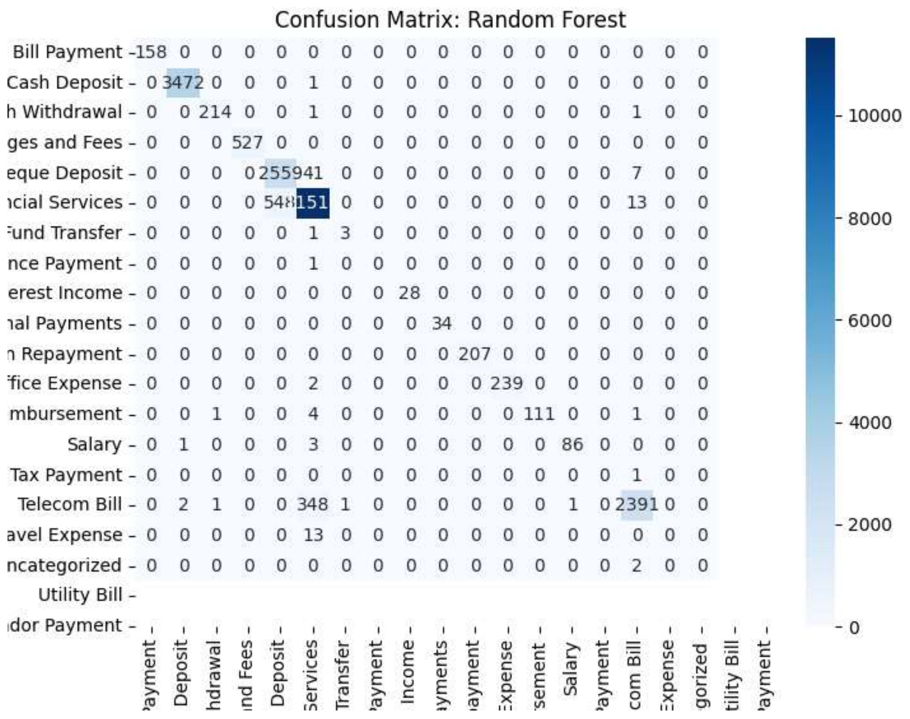
1. **Pipeline Creation**:
    - Each model was integrated into a machine learning pipeline that included:
        - TF-IDF vectorization for transforming TRANSACTION DETAILS into numerical features.
        - Model training on the transformed features.
2. **Hyperparameter Tuning**:
    - Grid search was conducted on Random Forest to optimize key parameters:
        - **Random Forest**: Explored n_estimators (50, 100, 200) and max_depth (None, 10, 20).
3. **Evaluation Metrics**:
    - **Accuracy**: Measures overall correctness of predictions.
    - **Precision and Recall**: Evaluated per class to account for imbalances in transaction tags.
    - **Weighted F1 Score**: Prioritized for model selection due to imbalanced data.

Models Evaluation

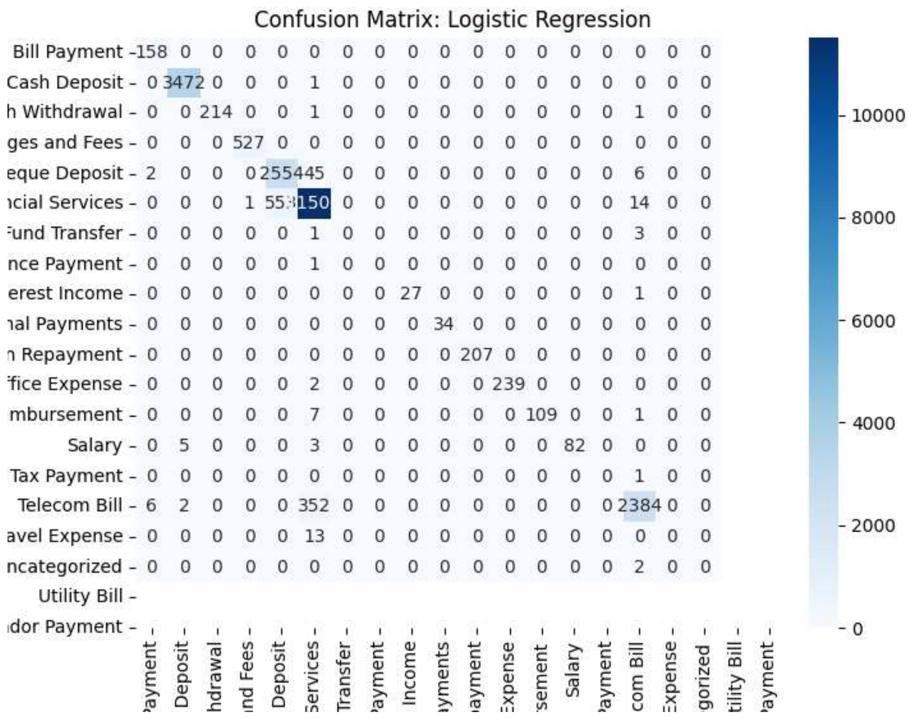
After training, all models were evaluated on the test set using the above metrics. The **Random Forest Classifier** outperformed other models with the highest weighted F1 score and balanced precision-recall values across most categories.

Confusion Matrices for each model:

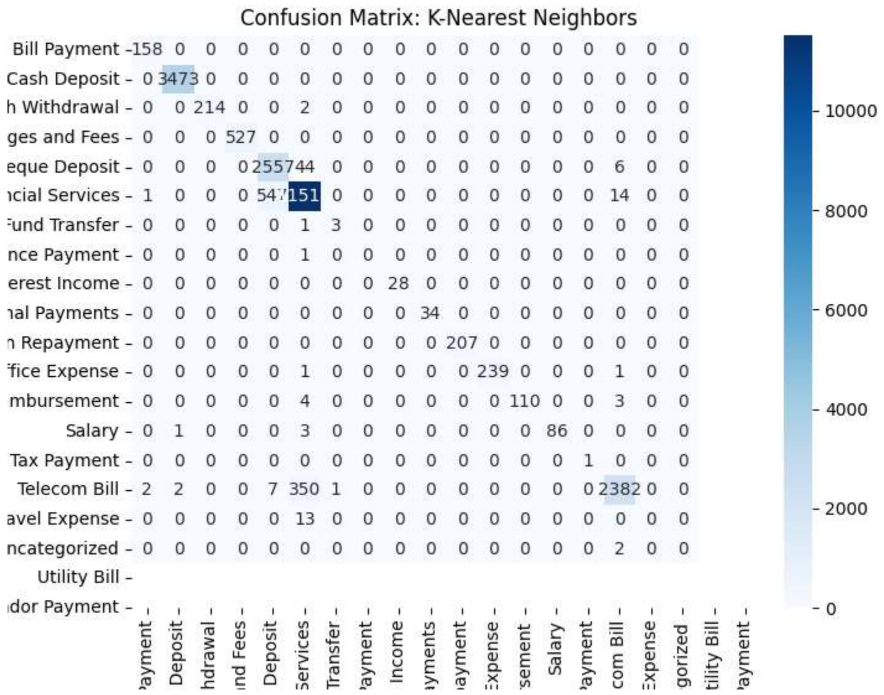
1. **Random Forest**:



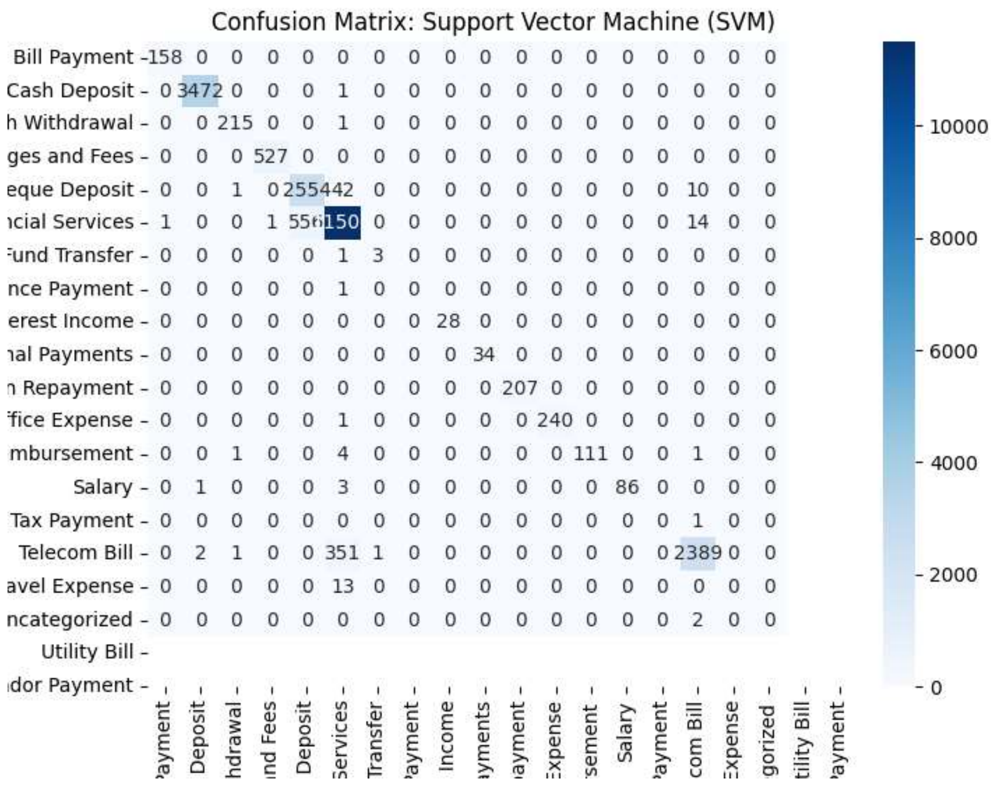
2. **Logistic Regression**:



3. **KNN**:


Confusion Matrix: K-Nearest Neighbors
Bill Payment
Cash Deposit
h Withdrawal
ges and Fees
eque Deposit
ıcial Services
Fund Transfer
nce Payment
erest Income
ıal Payments
ı Repayment
fice Expense
mbursement
Salary
Tax Payment
Telecom Bill
avel Expense
ncategorized
Utility Bill
ıdor Payment
10000
8000
6000
4000
2000
0

4. **SVM**:


Confusion Matrix: Support Vector Machine (SVM)
Bill Payment
Cash Deposit
h Withdrawal
ges and Fees
eque Deposit
ıcial Services
Fund Transfer
nce Payment
erest Income
ıal Payments
ı Repayment
fice Expense
mbursement
Salary
Tax Payment
Telecom Bill
avel Expense
ncategorized
Utility Bill
ıdor Payment
10000
8000
6000
4000
2000
0

Predicting Ledger Tags

The best model was saved in .pkl format with the help of joblib library in python. To ensure that we pre-process, the transaction details text, tfidf-vectorizer and label-encoder were also saved as .pkl files.

For predicting the ledger tags from raw data, following steps are taken:

1. Clean the data by handling null values.
2. Using the Tf-IDF vectorizer to pre-process the transaction details.
3. Data is sent to the saved model to get the predictions of ledger tags.
4. From the predictions, label-encoder is used to properly save the predicted label.

Challenges Faced

The main challenge faced was not enough information related to the reasons of transactions and notes related to them. This hinders the process of manual tagging as due to limited information and domain knowledge tags might be over simplified or complicated.

Conclusion

This project involved training machine learning model to figure out the ledger tags based on the transaction details provided in the dataset.

After performing following processes:

1. Data Cleaning
2. Feature Engineering
3. Manual Data Annotation
4. Model Training
5. Model Evaluation

The random forest model was the best performing model in comparison to the others.