ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY
COLLEGE OF ENGINEERING
DEPARTMENT OF SOFTWARE ENGINEERING

SOFTWARE COMPONENT DESIGN
Python ML Churn Prediction Model

Section C

Name	ID
Elbetel Shineda	ETS0406/13
Leul Mintesinot	ETS0766/13
Mihret Agegnehu	ETS0874/13
Mikiyas Bedasa	ETS0888/13
Natanim Ashenafi	ETS0979/13

Submitted to: Mr. Gizate Desalgn
Submission Date: December 18, 2024

Machine Learning Predicting Customer Churn

Introduction

This document provides a detailed explanation of the implementation, methodology, and logic behind the Python Machine Learning code for predicting customer churn. The code uses the Telco Customer Churn dataset to build a machine learning model that identifies customers likely to discontinue services.

Project Setup

Tools and Libraries Used

1. **pandas**: For data manipulation and preprocessing.
2. **numpy**: For numerical computations.
3. **matplotlib** and **seaborn**: For data visualization.
4. **scikit-learn**: For machine learning models and evaluation metrics.

Dataset

- **Source**: Telco Customer Churn dataset from Kaggle.
- **Size**: 7043 rows and 21 columns.
- **Target Variable**: `Churn` (Binary: Yes/No).
- **Features**: Includes customer demographics, service usage patterns, and account information.

Data Preprocessing

1. Handling Missing Values

- Columns like `TotalCharges` contained missing values due to empty strings.

Solution: Convert empty strings to NaN and replace NaN values with the mean of the column.
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

- df['TotalCharges'].fillna(df['TotalCharges'].mean(), inplace=True)

2. Encoding Categorical Variables

Categorical variables (e.g., `gender`, `Partner`, `Dependents`) were converted to numeric values using **Label Encoding** or **One-Hot Encoding** depending on their nature.
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

- df['gender'] = le.fit_transform(df['gender'])

3. Scaling Numerical Features

Features like `tenure` and `MonthlyCharges` were normalized using **Min-Max Scaling** to improve model performance.
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

- df[['tenure', 'MonthlyCharges', 'TotalCharges']] = scaler.fit_transform(df[['tenure', 'MonthlyCharges', 'TotalCharges']])

Exploratory Data Analysis (EDA)

1. Distribution of Churn

- The dataset has an approximate churn rate of **26.5%**.
- Visualized using a bar plot:
  sns.countplot(df['Churn'])

2. Correlation Analysis

- Heatmaps were generated to identify correlations between features and the target variable `Churn`.

  sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

3. Feature Relationships

- Histograms and box plots were created to analyze numerical features like `tenure` and `MonthlyCharges`.

Model Development

1. Splitting the Data

The dataset was split into **training** (80%) and **test** (20%) sets.

from sklearn.model_selection import train_test_split

X = df.drop('Churn', axis=1)

y = df['Churn']

- X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

2. Model Selection

- **Algorithms Tested**:
  - Logistic Regression
  - Random Forest
  - Gradient Boosting
  - K-Nearest Neighbors (KNN)
  - Support Vector Machines (SVM)
- KNN performed the best with an accuracy of **98.2%**.

3. Hyperparameter Tuning

Grid search was used to optimize hyperparameters for each model.

from sklearn.model_selection import GridSearchCV

param_grid = {'n_neighbors': [3, 5, 7], 'weights': ['uniform', 'distance']}
grid = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5)

- grid.fit(X_train, y_train)

Model Evaluation

Metrics Used

1. **Accuracy**: Proportion of correctly classified samples.
2. **Precision**: Proportion of positive predictions that are correct.
3. **Recall**: Proportion of actual positives that are correctly identified.
4. **F1-Score**: Harmonic mean of precision and recall.

Example of evaluation:
from sklearn.metrics import classification_report
y_pred = grid.best_estimator_.predict(X_test)
print(classification_report(y_test, y_pred))

Key Findings

- **Important Features**: `tenure`, `MonthlyCharges`, and `Contract` had the most influence on churn predictions.
- **Best Model**: KNN achieved the highest accuracy of **98.2%**.

Code Execution Instructions

Prerequisites

- Python 3.8 or higher.
- Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn.

Steps

1. Clone the repository:
   git clone https://github.com/mihretgold/churn_prediction_group2.git
2. Navigate to the project directory:
   cd churn-prediction
3. Run the notebook:
   jupyter notebook churn_prediction.ipynb

Conclusion

This documentation provides an in-depth explanation of the code implementation, from data preprocessing to model evaluation. The KNN model emerged as the best-performing algorithm, enabling accurate customer churn predictions to help businesses retain customers and minimize revenue loss.