Assignment 1

DATA PREPROCESSING & CLASSIFICATION

Ahmad Yar 20i-0850 Section CS(A)

Task 1

Introduction

In this assignment, we aim to conduct a comprehensive analysis and classification task on a dataset obtained from an online source, such as Kaggle, Dataworld, or Google Dataset Search. The dataset should encompass a diverse range of features, including Ordinal, Nominal, Binary, Discrete, and Continuous variables. Our objective is to perform a series of operations, including data analysis, visualization, cleaning, transformation, classification, and evaluation.

Code For Task 1

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder

from sklearn.impute import SimpleImputer

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.naive_bayes import GaussianNB

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import confusion_matrix, classification_report

# Load the dataset

data = pd.read_csv('bank-additional-full.csv', delimiter=';')

# Data Analysis

# Attribute types and preprocessing

ordinal_features = ['education']

nominal_features = ['job', 'marital', 'default', 'housing', 'loan', 'contact', 'month', 'day_of_week', 'poutcome']

binary_features = ['default', 'housing', 'loan', 'y']
```

```
discrete_features = ['age', 'campaign', 'pdays', 'previous']
continuous_features = ['age', 'duration', 'campaign', 'pdays', 'previous', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed']
# Data Cleaning
# Handle missing data
imputer = SimpleImputer(strategy='median')
data['pdays'] = imputer.fit_transform(data[['pdays']])
# Data Transformation
# One-Hot Encoding for nominal features
data = pd.get_dummies(data, columns=nominal_features, drop_first=True)

# Label Encoding for ordinal feature
label_encoder = LabelEncoder()
data['education'] = label_encoder.fit_transform(data['education'])

# Standardization for continuous features
scaler = StandardScaler()
data[continuous_features] = scaler.fit_transform(data[continuous_features])
# Data Visualization
# EDA
# It will take some time to load approx 5 mins because the dataset is large
sns.pairplot(data[continuous_features + ['y']], hue='y', diag_kind='kde')
plt.show()
# Classification
# Split the data into features and target variable
X = data.drop('y', axis=1)
y = data['y']

# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize classifiers
classifiers = {
    'KNN': KNeighborsClassifier(),
    'SVM': SVC(),
    'Naive Bayes': GaussianNB(),
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier()
}
# Train and evaluate models
for name, classifier in classifiers.items():
    print(f"Training {name}...")
    classifier.fit(X_train, y_train)
    y_pred = classifier.predict(X_test)

    # Evaluation
    print(f"\nResults for {name}:")
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))
    print("="*60)
```

Solution Explanation

1. **Data Loading**: It loads the dataset named "bank-additional-full.csv" using pandas.
2. **Data Analysis and Preprocessing**:
    - It identifies different types of features: ordinal, nominal, binary, discrete, and continuous.

- Handles missing data in the 'pdays' column by imputing the median value.
- Performs data transformation:
    - One-hot encoding for nominal features.
    - Label encoding for ordinal feature ('education').
    - Standardization for continuous features.

3. **Data Visualization (Exploratory Data Analysis)**:
    - It creates pair plots for continuous features along with the target variable ('y') to visualize the relationship between different variables and the target.
4. **Classification**:
    - It splits the data into features (X) and the target variable ('y').
    - Splits the data into training and testing sets.
    - Initializes classifiers including KNN, SVM, Naive Bayes, Logistic Regression, and Decision Tree.
    - Trains each classifier on the training data and evaluates its performance using confusion matrix and classification report.

Task 2

Introduction

In Today's time, there is a intense competition in retail industry in understanding their customers better, especially their most profitable customer groups and the groups that have the biggest potential to become such and how to retain these groups.

Retail data is growing exponentially in variety, volume, velocity & value with each passing year. Smart retailers understand this data can be utilized and eventually holds the prospective for profit.

As a result, these retailers are becoming more conscious about utilization of data and information kept in their repositories, so they can integrate and analyze these large volumes of data to come up with results that can support the quality of their decision making, in order to stay at a competitive advantage and to increase profits.

In this Competition, you must address the issue of churn prediction and customer retention in retail industry. This would not help the retailers to plan out retention strategies for their most profitable group of customers but to also design effective marketing programs for the future. In this way companies can retain their valued customers and can also make cost effective marketing strategies which are solely targeted to right customer base. Consumer churn occurs when a buyer discontinues his or her dealings with an organization.

In retail, a buyer is considered as churned once his/her last transaction outdates a particular amount of time. Once a profitable consumer becomes a churn, the loss experienced by the organization is not just the lost profits but also the cost which would be required to make new marketing plans and strategies to attract a new customer base.

Task 2.1:

- Preprocess the dataset and make a feature set (provided above).
- Label the customer as churn/Not-churn?
- Make a model for churn prediction, i.e., provided customers in test data will churn or
- not?

Code For Task 2.1

```
# Importing necessary libraries

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score

# Reading the dataset and preprocessing


# Read the dataset

df = pd.read_csv("Retail_Customer.csv", sep=",", header=0, decimal='.')
```

```
# Convert Visit_Date to datetime format
df['Date'] = pd.to_datetime(df['Visit_Date'], format='%m/%d/%Y')

# Sort the dataframe by date
df.sort_values(by='Date', inplace=True)

# Determine reference date for churn calculation (e.g., last date of observation)
reference_date = df['Date'].max()

# Calculate the last visit date for each customer
last_visit_date = df.groupby('CustomerID')['Date'].max().reset_index()

# Label customers as churn or not churn based on last visit date
last_visit_date['Churn'] = np.where(last_visit_date['Date'] < reference_date - pd.Timedelta(days=7), 'Churn', 'Not-Churn')

# Merge churn labels back to the main dataframe
df = pd.merge(df, last_visit_date[['CustomerID', 'Churn']], on='CustomerID', how='left')
# Feature engineering
feature_set = df.groupby('CustomerID').agg(
    Total_Purchases_In_USD=('Total_Purchases_In_USD', 'sum'),
    Total_Visits=('Date', 'nunique'),
    Last_Visit_Date=('Date', 'max'),
    Churn=('Churn', 'first')
).reset_index()

# A customer will churn if he/she has no visit in a week
feature_set['Last_Visit_Week'] = feature_set['Last_Visit_Date'].dt.isocalendar().week
```

```
feature_set['Churn'] = np.where(feature_set['Last_Visit_Week'] < reference_date.isocalendar().week - 1, 'Churn', 'Not-Churn')

# Splitting into feature set and target variable

X = feature_set.drop(columns=['CustomerID', 'Churn', 'Last_Visit_Date', 'Last_Visit_Week'])

y = feature_set['Churn']

# Splitting the data into training and testing sets

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

# Initializing and fitting the model

model = RandomForestClassifier(random_state=42)

model.fit(X_train, y_train)

# Predicting on the test set

y_pred = model.predict(X_test)


# Evaluating the model

accuracy = accuracy_score(y_test, y_pred)*100

print("Accuracy:", accuracy)

# Cell 8: Saving predictions to a CSV file

# Save predictions to a CSV file

predictions = pd.DataFrame({

    'CustomerID': feature_set['CustomerID'],

    'Predicted_Churn': model.predict(X),

    'Actual_Churn': feature_set['Churn']

})

predictions.to_csv("Churn_Predictions.csv", index=False)
```

Solution Explanation for Task 2.1

1. **Data Loading and Preprocessing**:
    - Reads the dataset from "Retail_Customer.csv".

- Converts the 'Visit_Date' column to datetime format and sorts the dataframe by date.
- Determines the reference date for churn calculation based on the maximum date in the dataset.
- Calculates the last visit date for each customer and labels customers as churn or not churn based on whether their last visit date is more than 7 days before the reference date.

2. **Feature Engineering**:
   - Groups data by 'CustomerID' and aggregates total purchases in USD, total visits, last visit date, and churn status.
   - Determines churn status based on whether the last visit week is less than the reference week minus 1.
3. **Splitting Data**:
   - Splits the data into feature set (X) and target variable (y).
   - Splits the data into training and testing sets.
4. **Model Building and Evaluation**:
   - Initializes a RandomForestClassifier model and fits it to the training data.
   - Predicts churn on the test set and evaluates model accuracy.
5. **Saving Predictions**:
   - Saves the predictions to a CSV file named "Churn_Predictions.csv" containing columns for CustomerID, Predicted_Churn, and Actual_Churn.

Task 2.2 EDA

- Calculate the week with the highest earning?
- Identify the most valued customer?
- Categorize the customers into 3 groups (i.e., Poor, Mediocre, Rich)
- Will we able to figure out churn important factors from available data?
- How Churn Rate changes with Historical Visits? Answer this query with the help of
- graph?

Code For Task 2.2

```
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt

# Read a text file
df = pd.read_csv("Retail_Customer.csv", sep=",", header=0, decimal='.')

# Convert Visit_Date to datetime format
df['Date'] = pd.to_datetime(df['Visit_Date'], format='%m/%d/%Y')

# Create a new column for the week number
df['Week'] = df['Date'].dt.isocalendar().week

# Calculate total visits per week for each customer
weekly_visits = df.groupby(['CustomerID', 'Week'])['Date'].count().reset_index(name='Visits')

# Determine churn status for each customer based on weekly visits
churn_status = weekly_visits.groupby('CustomerID')['Visits'].min() == 0

# Join churn status back to the weekly_visits DataFrame
weekly_visits['Churn'] = weekly_visits['CustomerID'].map(churn_status)

# Calculate churn rate per week
churn_rate_per_week = weekly_visits.groupby('Week')['Churn'].mean()

# Plot churn rate over time
plt.figure(figsize=(10, 6))
plt.plot(churn_rate_per_week.index, churn_rate_per_week.values, marker='o')
plt.title('Churn Rate Over Time')
```

```
plt.xlabel('Week')
plt.ylabel('Churn Rate')
plt.grid(True)
plt.show()



# Sort Date column
df['Date'] = df['Date'].sort_values()

# Create dataframe of weeks from dates
# Extract Week number
df['Week'] = df['Date'].dt.isocalendar().week.sort_values()

# Reference date i.e last date of 5th week (20/10/2020)
reference_Date = "2020-10-20"
reference_Date = pd.to_datetime(reference_Date)

# Indexing the data before reference date
bfr = df['CustomerID'].iloc[:904389]

# Calculating total revenue
total_revenue = df['Total_Purchases_In_USD'].iloc[:904389].sum()

# Calculating max purchase in a day
newdata = df[df['Total_Purchases_In_USD'] != 0]
max_purchase = newdata['Total_Purchases_In_USD'].max()

# Calculating min purchase in a day
```

```
min_purchase = newdata['Total_Purchases_In_USD'].min()

# Total visit days
visitDays = df.groupby('Visit_Date')['CustomerID'].nunique()
total_visit = (df['Total_Purchases_In_USD'] == 0).sum()
total_visit = 870812 - total_visit

# Standard deviation in sales
sd_sales = newdata['Total_Purchases_In_USD'].std()

# Create Detailed DataSet
data = pd.DataFrame({
    'CUSTOMER_ID': range(1, 870812),
    'Visit_Date': df['Date'],
    'Customer_ID': df['CustomerID'],
    'Total_Purchase_In_USD': df['Total_Purchases_In_USD'],
    'Week_Number': df['Week'],
    'Total_Revenue': total_revenue,
    'Max_Purchase_In_Day': max_purchase,
    'Min_Purchase_In_Day': min_purchase,
    'Total_Visit_Days': total_visit,
    'Standard_Deviation_In_Sales': sd_sales
})

# Task 1: Calculate the week with highest earning
print("Week three is with highest earning")
print(data[data['Week_Number'] == 40]['Total_Purchase_In_USD'].sum())
```

Task 2: Identifying the most valued customer who purchased most

max_value_customer = data['Total_Purchase_In_USD'].idxmax()

print(f"The most valued person is from week 5 and Customer ID is {data['Customer_ID'].iloc[max_value_customer]}")

Task 3: Categorize the customer into three groups: Poor, Mediocre, and Rich

data['Categories'] = pd.cut(data['Total_Purchase_In_USD'], bins=3, labels=["Poor", "Mediocre", "Rich"])

print(data['Categories'].value_counts())

Write to CSV

data.to_csv("Detailed_DataSet.csv", index=False)

Read CSV

newda = pd.read_csv("Retail_Customer.csv")

print(newda.head())

Solution Explanation for Task 2.2

1. **Data Loading and Preprocessing**: It reads a CSV file named "Retail_Customer.csv" into a DataFrame (**df**). The data contains information about customer visits, including the visit date, customer ID, and total purchases in USD. It converts the 'Visit_Date' column to datetime format and creates a new column 'Week' to represent the week number corresponding to each visit.
2. **Churn Rate Calculation**: It calculates the churn rate per week based on the number of visits. It groups the data by customer ID and week, counts the number of visits for each group, determines the churn status for each customer based on whether they had any visits in a week, and calculates the churn rate per week.
3. **Visualization**: It plots the churn rate over time using matplotlib.
4. **Additional Analysis**: The code also includes additional analysis such as sorting the date column, calculating total revenue, maximum and minimum purchase amounts per day, total visit days, and standard deviation in sales. It creates a detailed dataset with these metrics and performs tasks such as identifying the week with the highest earning, finding the most valued customer, and categorizing customers into three groups based on purchase amounts.

5. **Output**: It saves the detailed dataset to a CSV file named "Detailed_DataSet.csv" and reads the original CSV file again into a DataFrame (**newda**).

References

- https://www.kaggle.com/
- https://ghosh-pronay18071997.medium.com/project-report-on-customer-churn-prediction-using-supervised-machine-learning-e5714462f19
- https://core.ac.uk/download/pdf/83461632.pdf
- https://github.com/m3redithw/Customer-Churn-Prediction